

# **ADAPTIVE PREDICTIVE CODING OF SPEECH USING MAXIMUM ENTROPY METHOD**

**A Thesis Submitted  
in Partial Fulfilment of the Requirements  
for the Degree of  
MASTER OF TECHNOLOGY**

**by  
V. S. MAHALINGAM**

**to the  
DEPARTMENT OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY KANPUR  
JUNE, 1982**

1 JUN 1984

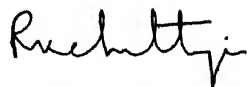
CEN. ~~ARY~~  
Acc. No. A 82646

EE-1982-M-MAH-ADA

82646

## CERTIFICATE

This is to certify that the thesis entitled 'ADAPTIVE PREDICTIVE CODING OF SPEECH USING MAXIMUM ENTROPY METHOD' by V.S. Mahalingam has been carried out under our supervision and that it has not been submitted elsewhere for a degree.



( P.K. Chatterjee )  
Professor



( K.R. Sarma )  
Professor

Department of Electrical Engineering  
Indian Institute of Technology  
Kanpur 208016, India

## ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my thesis supervisors Dr. K.R. Sarma and Dr. P.K. Chatterjee for their wise counsel and encouragment. I have been benefitted immensely from the discussions, I had with them from the time the project was conceived, through its execution upto the time of thesis completion.

I am thankful to Dr. V.P. Sinha and Dr. S.K. Mullick for useful discussions on digital filters and MEM respectively.

I am grateful to my service, the Defence Research and Development Organization (DRDO), for having provided me the opportunity to work towards my M.Tech. degree at I.I.T., Kanpur

My wife and daughter deserve a word of special praise, for having donated many evenings and Sundays, which, in my opinion rightly belong to them, for the sake of my 'knowledge seeking pursuit'.

My thanks are also due to Mr. C.M. Abraham of ACES, for the efficient typing, from an almost illegible manuscript.

V.S. Mahalingam



When using merely ones and zips  
To carry sound from the lips  
The leading issue always fits  
'IN A SECOND - HOW MANY BITS?'  
That's the question we address  
Lower bit rate is the stress  
The rate depends upon the trade  
Of bits and bucks for service grade

'M.R. Aaron', in

Guest editorial, IEEE Trans. Commun.  
vol. COM-30, No.4, April 1982.

## TABLE OF CONTENTS

	Page
LIST OF FIGURES	viii
LIST OF TABLES	viii
ABSTRACT	ix
CHAPTER 1 INTRODUCTION	1
1.1 Historical background	1
1.2 Problem formulation	5
1.3 Organization of thesis	6
REFERENCES	8
CHAPTER 2 LINEAR PREDICTION	
2.1 Linear predictive filter	9
2.2 Autoregressive modelling	14
2.3 Estimation of predictor coefficients	16
2.4 Method of compute the parameters	18
2.5 Filter stability	20
2.6 Optimum number of poles	21
Results	23
REFERENCES	25
CHAPTER 3 MAXIMUM ENTROPY METHOD	
3.1 Statistical definition of information	27
3.2 Relationship between MEM estimation and linear predictive filtering	28
3.3 Determination of autocorrelation function by MEM	30

3.4 Stability	37
3.5 Order of MEM prediction filter	38
Results	39
REFERENCES	41
CHAPTER 4 SYSTEM DESCRIPTION	
4.1 Optimum quantization	43
4.2 General differential quantizer- predictor	47
4.3 Performance measures	50
4.4 Adaptive quantization	52
4.5 Adaptive prediction	55
4.6 Transmission of parameters	58
Results	60
REFERENCES	73
CHAPTER 5 VARIABLE LENGTH HUFFMAN CODING	
5.1 Introduction	76
5.2 Huffman code	78
5.3 Code construction and buffer management	80
Results	83
REFERENCES	85
CHAPTER 6 CONCLUSION	
6.1 Concluding remarks	86
6.2 Suggestions for future work	88
REFERENCES	91

## APPENDIX A

A.1	Signal flow description	95
A2.1	Gaussian number generation	95
A2.2	Speech samples	96
A.3	Digital filter	97

## APPENDIX B

107

## LIST OF FIGURES

Fig.No.		Page
1.1	Basic DPCM	2
2.1	Forward prediction filter	10
3.1	Lattice filter	36
4.1	General block diagram of DPCM coder-decoder	48
A.1	Signal flow diagram	94
A.2	Flow chart (FILORD)	101
A.3	Flow chart (POZE)	102
A.4	Flow chart (FILCON)	104
A.5	Flow chart (SAMFIL)	106

## LIST OF TABLES

Table No.		Page
4.1	Optimum quantizer for Laplacian density	46

## ABSTRACT

Speech, due to its inherent redundancy, lends itself readily to the method of predictive coding. By employing a linear predictor and MMSE criterion, the DPCM loop is studied with adaptation of quantizer and predictor. The autocorrelation coefficients, for updating the linear predictor tap coefficients are determined by maximum entropy method, a more accurate method for power spectrum evaluation than the conventional windowing procedures.

With variable rate Huffman coding, the quantizer output is further compressed in bandwidth. It is found that nearly 17.9 Kb/s and 12.8 Kb/s are needed as final data rate for 4 bit and 3 bit quantizers respectively. In addition, 1-2 Kb/s is needed for parameters transmission from coder to decoder.

The study is simulated on the DEC-10 computer of I.I.T., Kanpur.

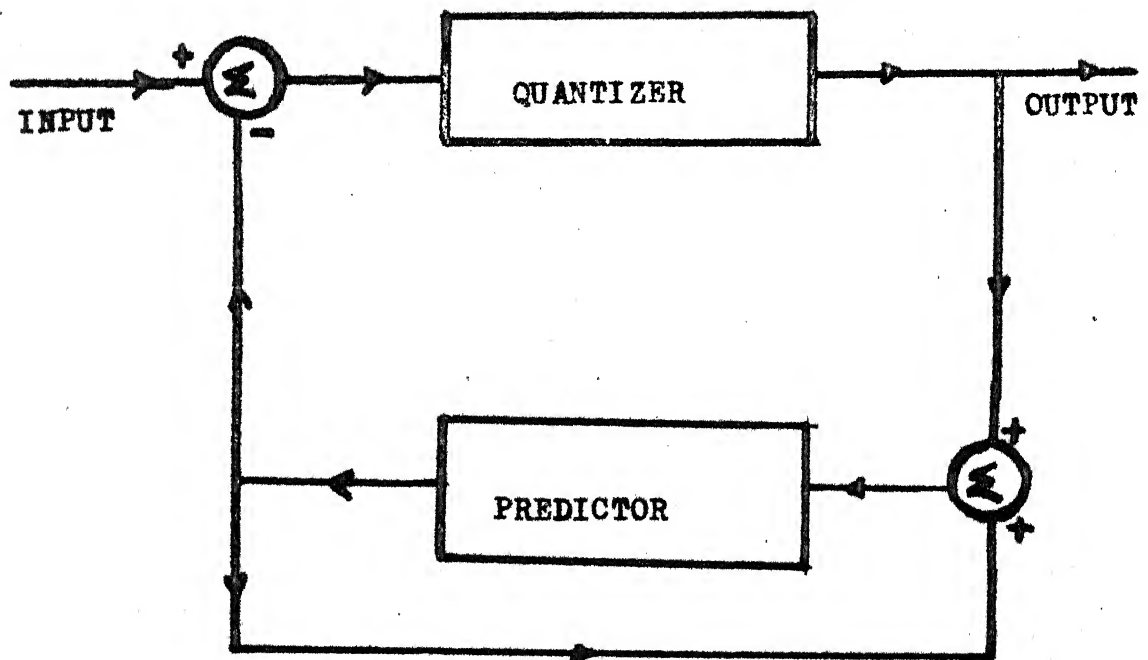
## CHAPTER 1

### INTRODUCTION

#### 1.1 HISTORICAL BACKGROUND

The purpose of speech is effective communication. Nearly 90 percent of the processed signal in communication is speech. This enormous figure creates an ever increasing demand on the existing digital facilities (e.g. communication channels and data storage), which can be alleviated to great extent by representing the same amount of information with fewer bits at the expense of more sophisticated processing. The method by which this is achieved goes by various names of 'redundancy reduction', 'data compression' and others. Effectively redundancy could be used to decrease bandwidth, to increase speed, to lower probability of error or to lower average signal power [1].

Historically, redundancy reduction was investigated in early 1950's by Oliver, Elias and others. Each of these papers relied heavily on the theory of prediction developed by Wiener in the early 1940's. According to Wiener [2], any operation is to be based on the data available alone, and no other additional apriori knowledge is available. That is, the prediction is to be based on the past values of input only.



**Fig. 1.1 BASIC DPCM**

---



Speech and picture signal, due to their inherent redundancy present, lend themselves readily to the reduction techniques of predictive coding. Instead of sending the quantized sample directly like in PCM, if only the variations between successive samples could be sent and at the receiver from these variations that are received the original highly correlated signal with its redundancy could be reconstructed, a reduction in data rate is achieved. This is the basic principle of all differential coders.

Differential pulse code modulation (DPCM) was one of the early versions of differential coders to be studied. According to NS Jayant 'Simple DPCM is still one of the classic triumphs of intuitional waveform coding' [3]. A DPCM system is shown in Fig. 1.1.

The predictor, based on past samples, forecasts what is the present sample. This predicted part is subtracted from the input, and the transmitted signal is the unpredictable component. The predictor parameters are determined by different methods. Various criteria are employed towards making the predictor ideal (i.e.) in making the unpredictable component which is prediction error tend towards zero.

Each method of error minimisation is assigned a 'cost'. The standard cost functions are the following :

- i) Cost function  $C(e)$  of error is equal to the squared error

$$C(e) = e^2 \quad (1.1)$$

ii) Cost-function  $C(e)$  is equal to the modulus of error

$$C(e) = |e| \quad (1.2)$$

iii) When over a small range centred around origin  $C(e) = 0$  and in the remaining region equals one

$$C(e) = \begin{cases} 0 & |e| \leq \frac{\delta}{2} \\ 1 & |e| > \frac{\delta}{2} \end{cases} \quad (1.3)$$

The first method of error minimisation goes by the name Minimum Mean Squared Error (MMSE) criterion. In this method, the estimate of the present sample is always the mean of the a posteriori density (conditional mean) [4]. The second approach, known as the Maximum a posteriori estimation (MAP) gives the estimate value as the one at which the a posteriori density is maximum. When the a posteriori density is symmetrical about the conditional mean, the optimum estimate is the conditional mean.

The prediction process can be either linear or non-linear [5]. Linear prediction is easily amenable to mathematical formulations and gives closer insight into the problem. Under the linear prediction case, the error at time 'n' is uncorrelated with the input at every point of the observation interval. That is they are orthogonal. For most of the analysis purposes, the choice for input is Gaussian samples. Speech with some approximations can be modelled Gaussian. For linear prediction with Gaussian input,

MMSE and MAP estimates coincide. For Gaussian assumption of input, the optimum linear predictor for minimising mean squared error is the best of any type. In other words, a nonlinear processor cannot give an estimate with a smaller mean squared error. Because of all these observations, in this thesis linear prediction and MMSE criterion are used.

## 1.2 PROBLEM FORMULATION

The DPCM system shown in Fig. 1.1 is simulated on DEC-10 computer and studied in this thesis. This involves the following :

- i) Generation of a sequence of Gaussian numbers to be used as input
- ii) Design of a lowpass digital filter to bandlimit input and decoded output
- iii) Determination of the predictor coefficients by autocorrelation method (explained in Chapter 2). Maximum entropy method is used (explained in Chapter 3) for determining the autocorrelation coefficients
- iv) Adaptation of the quantizer to match the changes in variance of the input signal
- v) Adaptation of the predictor to match the changes in the short term spectrum between adjacent segments due to the nonstationarity of speech
- vi) Determination of statistics of probability distribution of occupation of different quantizer levels

- vii) Binary coding of the output of quantizer
- viii) Construction of variable length Huffman codes to conserve final data rate.

The above operations (i) to (viii) pertain to the coder side. Corresponding inverse operations are done at the receiver to reconstruct the signal back. This necessitates the transmission of periodic predictor coefficients and the segment standard deviation to the receiver.

### 1.3 ORGANIZATION OF THESIS

After the introductory chapter dealing with historical background and problem formulation, Chapter 2 explains linear prediction. Also included are the methods to calculate the solutions and determination of filter stability. Chapter 3 deals with maximum entropy method (MEM) and how MEM is used to determine the autocorrelation coefficients. Basic theoretical explanations pertaining to the statistical modelling of speech, optimum quantization, adaptation of quantizer and predictor forms the material of Chapter 4. Concept of variable length Huffman codes and the construction of the same is covered in the 5th chapter. Conclusions are drawn in Chapter 6. Also pointed out are the possible future areas of work, pertaining to the thesis problem. Apart from the above six chapters are included two appendices A and B.

Starting with signal flow diagram, Appendix A deals with the generation of Gaussian random numbers and the design of the digital LPF. This design is explained in a logical manner via flowcharts. Appendix B is the DECFORTRAN-10 program listing of the simulation experiment. Ample comment statements are included to facilitate ease of following the program.

## REFERENCES

- [1] R.W. Lucky, 'Adaptive redundancy removal in data transmission', BSTJ, vol. 47, No.4, pp 549-573.
- [2] N. Wiener, The extrapolation, interpolation and smoothing of stationary time series with engineering applications, Cambridge, Mass. M.I.T. Press, 1949.
- [3] N.S. Jayant (ed)., Waveform quantization and coding, New York, IEEE Press, 1976.
- [4] H.L. VanTrees, Detection, Estimation and Modulation Theory - Part I, New York, John Wiley and Sons, 1968.
- [5] A.V. Balakrishnan, 'An adaptive nonlinear data predictor', Proc. National Telemetry Conf., pp 6.5.1 - 6.5.15, 1962.

## CHAPTER 2

### LINEAR PREDICTION

#### 2.1 LINEAR PREDICTIVE FILTER

One of the most powerful speech processing techniques is the method of linear prediction. For redundancy removal to curb data rate, (bandwidth) predictive coding has emerged as a powerful tool. It is applied not only in speech but in image processing also.

The basic idea is to predict what would be the present sample based on the knowledge of previous 'p' samples where  $1 \leq p \leq n-1$ . As mentioned in Section 1.1 we assume a linear prediction and MMSE as the criterion.

The analysis is approached from the principle of orthogonality. If  $x_1, x_2 \dots x_N$  are the input (speech) data samples, an estimate of a random variable  $y$  by a linear combination is [1]

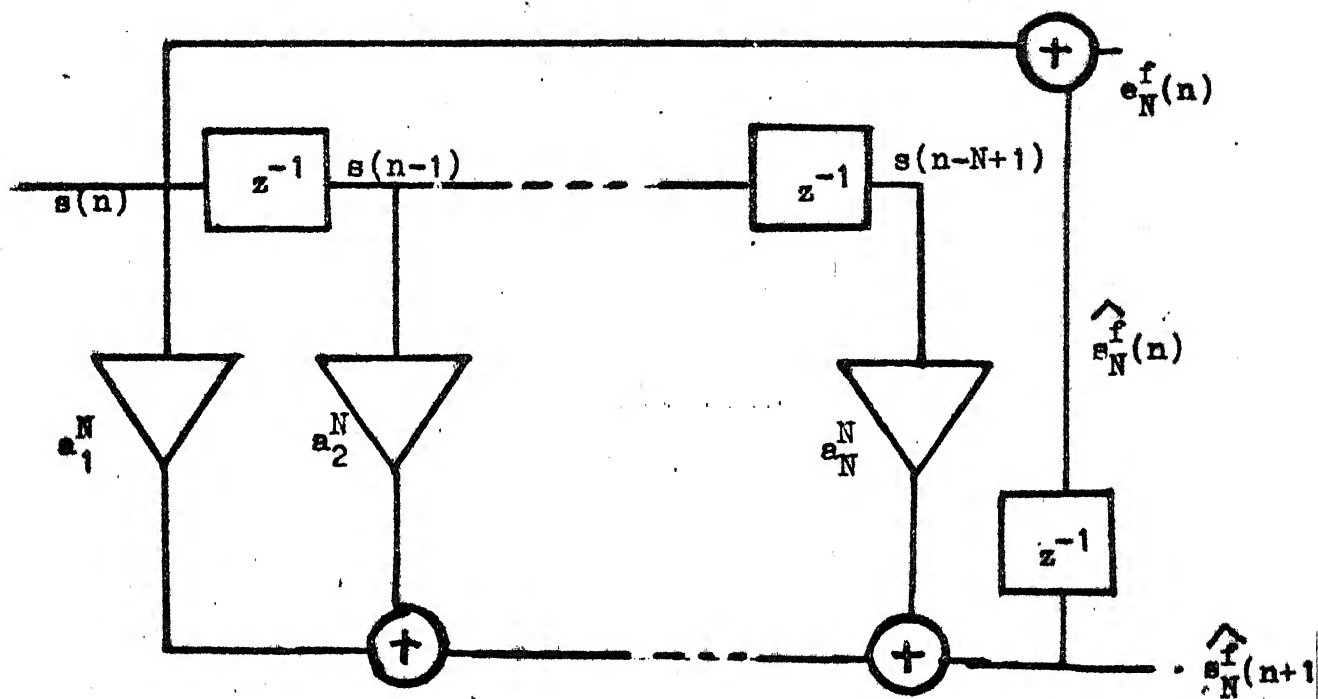
$$\hat{y} = a_1 x_1 + a_2 x_2 \dots + a_N x_N \quad (2.1)$$

The mean squared error is

$$P = \epsilon[y - \hat{y}]^2 \quad (2.2)$$

where  $\epsilon(\cdot)$  is the expectation operation.

This error is minimum when the estimation error



**Fig. 2.1 Forward prediction filter**

$\hat{s}_N^f(n)$  Predictor of  $s(n)$

$e_N^f(n)$  Predictor error



is orthogonal to the data  $x_k$ ,  $k = 1, \dots, N$  [2], i.e.

$$\varepsilon(ex_k) = 0 \quad (2.4)$$

Then the resulting mean squared error is

$$P = \varepsilon(e^2) = \varepsilon(ey) \quad (2.5)$$

Expanding these results to estimate the value of  $s(n+1)$  of a random signal, we get (assuming the random signal as  $s(n)$ )

$$\hat{s}_N^f(n+1) = a_1^N s(n) + \dots + a_N^N s(n-N+1) \quad (2.6)$$

$$= \sum_{k=1}^N a_k^N s(n-k+1) \quad (2.7)$$

This involves the  $N$  most recent values of  $s(n-k)$ .

$$e_N^f(n) = s(n) - \hat{s}_N^f(n) \quad (2.8)$$

The coefficients  $a_k^N$  that minimize the MS value of the prediction error defines an FIR filter as shown in Fig.2.1. The superscripts  $f$  and  $N$  denote the forward error filter and its order respectively.  $e_N^f(n)$  is called the forward prediction error. By equation (2.4)

$$\varepsilon[e_N^f(n) s(n-k)] = 0 \quad 1 \leq k \leq N \quad (2.9)$$

This yields the system in terms of the autocorrelation coefficients as a set of linear equations

$$\begin{aligned}
r(0)a_1^N + r(1)a_2^N + \dots + r(n-1)a_N^N &= r(1) \\
r(1)a_1^N + r(0)a_2^N + \dots + r(n-2)a_N^N &= r(2) \\
\vdots & \\
r(n-1)a_1^N + r(n-2)a_2^N + \dots + r(0)a_N^N &= r(n)
\end{aligned} \tag{2.10}$$

Replacing  $(n+1)$  by  $n$  in equation (2.6)

$$\hat{s}_N^f(n) = a_1^N s(n-1) + \dots + a_N^N s(n-N) \tag{2.11}$$

$$= \sum_{k=1}^N a_k^N s(n-k) \tag{2.12}$$

The error  $e_N^f(n)$  is

$$e_N^f(n) = s(n) - \sum_{k=1}^N a_k^N s(n-k) \tag{2.13}$$

As shown in Fig. 2.1 the input is  $s(n)$  and output is  $e_N^f(n)$ .

Taking  $z$  transforms, the system transfer function

$$H_N^f(z) = \frac{e_N^f(z)}{s(z)} = 1 - \sum_{k=1}^N z \text{ of } [a_k^N s(n-k)/s(n)] \tag{2.14}$$

$$= 1 - \sum_{k=1}^N a_k^N z^{-k} \tag{2.15}$$

$$\text{Since } s(n-k) = z^{-k} s(n). \tag{2.16}$$

If the signal  $s(n)$  could be estimated from the recent past 'N' values, equally well, it could be estimated from 'N' closest future values. This gives rise to the concept of

Backward error filter. By a similar approach as in forward error filter, we get

$$\sum_{r_1=1}^N b_{r_1}^N r^{(1-k)} = r(k) ; \quad 1 \leq k \leq N \quad (2.17)$$

This is identical to the set of equations (2.10). So,

$b_{r_1}^N = a_k^N$  and the backward predictor of  $s(n)$  is

$$\hat{s}_N^b(n) = a_1^N s(n+1) + \dots + a_N^N s(n+N) \quad (2.18)$$

The superscript 'b' denotes that it refers to the backward error filter. The backward error  $e_N^b(n)$  is

$$e_N^b(n) = s(n) - \sum_{k=1}^N a_k^N s(n+k) \quad (2.19)$$

It is seen that  $e_N^f(n) = e_N^b(n)$ . So, by taking 'z' transforms the system transfer function  $H_N^b(z)$  is

$$H_N^b(z) = 1 - \sum_{k=1}^N a_k^N z^k \quad (2.20)$$

It is seen

$$H_N^b(z) = H_N^f(1/z) \quad (2.21)$$

The mean squared error denoted by  $E_N^f$  and  $E_N^b$  for the forward and backward error filters respectively are

$$E_N^f = r(0) - \sum_{k=1}^N a_k^N r(k) = E_N^b \quad (2.22)$$

## 2.2 AUTOREGRESSIVE MODELLING

An autoregressive process (AR) of order  $M$  is a random signal  $s(n)$  satisfying the recursion equation

$$s(n) - c_1 s(n-1) - \dots - c_M s(n-M) = \eta(n) \quad (2.23)$$

where  $\eta(n)$  is stationary white noise with autocorrelation and power spectral density as

$$r_{\eta\eta}(m) = \sigma^2 \delta(m) ; \quad S_{\eta\eta}(z) = \sigma^2 \quad (2.24)$$

It is seen from the definition that  $s(n)$  is the output of a linear system with input  $\eta(n)$  and system function  $T(z)$

$$T(z) = \frac{1}{1 - \sum_{k=1}^M c_k z^{-k}} \quad (2.25)$$

If the system is stable then  $s(n)$  is a stationary process represented by the convolution

$$s(n) = \sum_{r=0}^{\infty} h(r) \eta(n-r) \quad (2.26)$$

where  $h(n)$  is the causal inverse transform of  $T(z)$ .

This shows that for any  $k \geq 1$ ,  $s(n-k)$  is a linear combination of only the past values of  $\eta(n)$ . So,

$$\varepsilon[s(n-k) \eta(n)] = 0 ; \quad k \geq 1 \quad (2.27)$$

Since  $\eta(n)$  is white noise by assumption.

The predicted value  $\hat{s}_N^f(n)$  of  $s(n)$  of order  $N \geq M$  is

$$\hat{s}_N^f(n) = c_1 s(n-1) + c_2 s(n-2) + \dots + c_M s(n-M) \quad (2.28)$$

$$\text{The predictor error } \hat{e}_N(n) = s(n) - \hat{s}_N^f(n) = \eta(n) \quad (2.29)$$

This is seen from equation (2.23) also. The resulting error is  $\epsilon[e_N^2(n)] = \sigma^2$ .

The equation (2.28) is of the form of equation (2.7)

where

$$a_k^N = \begin{cases} c_k & 1 \leq k \leq M \\ 0 & M < k \leq N \end{cases} \quad (2.30)$$

Now with  $H_N^f(z)$  defined as  $1 - \sum_{k=1}^N a_k^N z^{-k}$ , it is seen

that  $T(z)$  of the AR process is specified as

$$T(z) = \frac{1}{H_N^f(z)} \quad (2.31)$$

So, clearly  $s(n)$  is the output of  $T(z)$  with input  $\eta(n)$ .

If,  $S(z)$  is the output power spectrum, then

$$S(z) = \frac{S_{\eta\eta}(z)}{H_N^f(z) H_N^f(1/z)} \quad (2.32)$$

Since the numerator is  $\sigma^2$ , it can be written

$$S(\omega) = S(e^{j\omega T}) = \frac{\sigma^2}{\left| 1 - \sum_{k=1}^M c_k e^{-jk\omega T} \right|^2} \quad (2.33)$$

It is seen that by predictor, we could remove the predictable portion which on subtraction gives out white noise  $\eta(n)$ .

The AR process, with input as  $\eta(n)$  is able to get back  $s(n)$ . For these, the knowledge of coefficients  $a_k^N = c_k$ ,  $k = 1, 2, \dots$  is essential which is done in the following section.

### 2.3 ESTIMATION OF PREDICTOR COEFFICIENTS

From equation (2.10) it is seen that the basic necessity is to know the tap coefficients  $a_k^N$ ,  $k = 1, \dots, M = c_k$ . MMSE criterion is employed to determine them. The total squared error is denoted by  $E [3] = E_N^f = E_N^b$ ,

$$E = \sum_{n=n_0}^{n_1} [e_N^f(n)]^2 \quad (2.34)$$

$$= - \sum_{n=n_0}^{n_1} \left[ \sum_{k=0}^M c_k s(n-k) \right]^2 ; c_0 = -1 ; \quad (2.35)$$

$$= - \sum_{n=n_0}^{n_1} \sum_{k=0}^M \sum_{i=0}^M c_k s(n-k) s(n-i) c_i \quad (2.36)$$

where  $n_0$  and  $n_1$  define the index limits over which error minimization occurs.

Defining

$$c_{ki} = \sum_{n=n_0}^{n_1} s(n-k) s(n-i) \quad (2.37)$$

The total squared error  $E$  is

$$E = - \sum_{k=0}^M \sum_{i=0}^M c_k c_{ki} c_i \quad (2.38)$$

Equation (2.38) shows that the total squared error is in quadratic form (i.e.) at most powers of two in the coefficients are obtained. Minimization of  $E$  is obtained by setting the partial derivative of  $E$  with respect to  $c_k$ ;  $k = 1, \dots, M$  to zero and solving. Therefore,

$$\frac{\partial E}{\partial c_k} = 0 = -2 \sum_{k=0}^M c_k c_{ki} \quad (2.39)$$

Since  $c_0 = -1$

$$\sum_{k=1}^M c_k c_{ki} = -c_{0k} \quad k = 1, 2, \dots, M \quad (2.40)$$

The  $M$  unknown predictor coefficients are obtained by solving this set of  $M$  linear simultaneous equations known as normal equations. It is seen that data  $s(n)$  from  $(n_0 - M)$  to  $n_1$  are needed.

Assuming that a sequence of  $N$  speech samples  $[s(n)] = [s(n-1), s(n-2) \dots s(0)]$  is available, the covariance method is defined by setting  $n_0 = M$  and  $n_1 = N-1$  so that the error is minimized only over the interval  $[M, N-1]$  and all  $N$  samples are used in calculating the covariance matrix elements  $c_{ki}$ .

The autocorrelation method is defined by setting  $n_0 = -\infty$  and  $n_1 = +\infty$  and then defining  $s(n) = 0$  for  $n < 0$  and  $n \geq N$ . These limits allow  $c_{ki}$  as

$$c_{ki} = \sum_{n=-\infty}^{\infty} s(n-k) s(n-i) \quad (2.41)$$

$$= \sum_{n=-\infty}^{\infty} s(n) s(n+|k-i|) \quad (2.42)$$

$$= \sum_{n=0}^{N-1-|k-i|} s(n) s(n+|k-i|) = r(|k-i|) \quad (2.43)$$

Thus, although error is minimized over an infinite interval, equivalent results are obtained by minimizing only over  $[0, N+M-1]$  as  $s(n)$  is truncated to zero for  $n \geq N$  and  $< 0$  [4].

The name covariance method is drawn from control theory literature where for a zero mean signal  $c_{ki}$  defines the  $k$ th row and  $i$ th column element of a covariance matrix. Autocorrelation method draws its name from the fact that  $c_{ki}$  reduces to the definition of short term autocorrelation at the delay  $|k-i|$ . It needs to be emphasized that these terms are not based upon the standard usage of the covariance function as the correlation function with means removed.

## 2.4 METHODS TO COMPUTE THE PARAMETERS

Employing either covariance or autocorrelation methods means solving a set of  $M$  linear equations in  $M$  unknowns. Various standard methods exist for this [5]. Some of them are :

Gauss elimination method,

Crout reduction method, and



The autocorrelation matrix, being symmetric and Toeplitz in nature lends itself to some special methods of solution. The set of equations can be written as

$$\begin{bmatrix} r(0) & r(1) & \dots & r(M-1) \\ r(1) & & & \\ \vdots & & & \\ r(M-1) & & & \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} r(1) \\ \vdots \\ r(M) \end{bmatrix} \quad (2.44)$$

$= A X = B$

where A is MxM size,  $X = M \times 1$  size; and B is Mx1 size.

To solve this matrix equation (2.44) an elegant recursive procedure was derived by Levinson [6]. This method was then reformulated by Robinson [7]. Levinson's method assumed the column vector 'B' to be a general vector. For the autocorrelation case, it is seen that the elements of the vector 'B' are those elements already present in 'A'. Making use of this fact, Durbin [8] devised a faster method. The recursive procedure of Durbin is given below.

$$\text{Error } e_0 \text{ at the start} = r(0) \quad (2.45)$$

for values of i from 1, 2, ... M

$$k_i = -[r(i) + \sum_{j=1}^{i-1} c_j^{(i-1)} r(i-j)]/e_{i-1} \quad (2.46)$$

$$c_i^{(i)} = k_i \quad (2.47)$$

$$c_j^{(i)} = c_j^{(i-1)} + k_i c_{i-j}^{(i-1)} \quad 1 \leq j \leq i-1 \quad (2.48)$$

$$e_i = (1 - k_i^2) e_{i-1} \quad (2.49)$$

where the superscript  $i$  indicates the iteration number.

The final coefficients are

$$c_j = c_j^{(M)} \quad 1 \leq j \leq M \quad (2.50)$$

The intermediate quantities  $k_i$  ;  $1 \leq i \leq M$  are known as reflection coefficients or partial correlation (PARCOR) coefficients as they are the correlation coefficients of the branches when the feed forward and feed back filters are put in the lattice structure. The minimum total error  $e_i$  is evaluated at each step. This is always positive and it decreases (or remains the same) with increasing  $M$ .

## 2.5 FILTER STABILITY

The feed forward prediction filter is stable if all its roots lie inside the unit circle (i.e.)

$H_N^f(z) = 1 - \sum_{k=1}^N a_k^N z^{-k}$  is a Hurwitz polynomial with all roots  $z_i$  lying inside the unit circle.

When the autocorrelation coefficients are positive definite [1] (which is assured when they are calculated from a nonzero signal) the solution of the autocorrelation method gives

predictor coefficients that guarantee that all roots lie inside the unit circle [3]. But sometimes, round-off errors can cause the autocorrelation matrix to be ill-conditioned. Then the check for stability becomes essential. While recursively evaluating the Durbin's algorithm, the intermediate parameters so calculated give much insight into the stability criterion. At every iteration the error  $e_i$  is checked whether it is positive, i.e.

$$e_i > 0; \quad 1 \leq i \leq M \quad (2.51)$$

This actually means that the PARCOR coefficients  $k_i$   $1 \leq i \leq M$  must have absolute value less than unity, i.e.

$$|k_i| < 1 \quad 1 \leq i \leq M \quad (2.52)$$

This is the necessary and sufficient condition for the filter to be stable [4].

## 2.6 OPTIMUM NUMBER OF POLES

The filter order  $M$  determines the number of poles that are present in the filter transfer function. The minimum value of  $M$  that is necessary to arrive at a particular required value of prediction error is chosen. This is dependent on many factors. In the autocorrelation method, it is seen the coefficients of autocorrelation are solved in a set of linear equations to get the predictor parameters whose number is  $\leq M$ . If, say after a lag of  $\tau$ , the

correlation becomes negligible such that even if one takes into account values beyond  $\tau$  and solves, it is found that the tap coefficients of the predictor are almost zero. For example, for filter order of  $M_1 < M$ , we may find that the filter coefficients almost taper to zero. In this case, nothing much is gained by taking an order of filter  $> M_1$ .

In the context of speech analysis, studies by various authors have established that the optimum number of poles for the filter is three [9], [10]. The plot of predictor gain versus number of poles, almost saturate after  $M \geq 3$ . N.S. Jayant reports the same phenomenon for  $M \geq 2$  and he gives results of simulation studies with a first order predictor [11].

In this thesis, therefore, the order of the filter is maintained at three. The simulation studies are carried out for values of  $M = 1, 2$ , and 3.

THE LINEAR FILTER COEFFICIENTS  $c_1, c_2$  and  $c_3$  (FOR 3rd ORDER FILTER) FOR SPEECH SAMPLES AS INPUT

Block No.	$c_1$	$c_2$	$c_3$
1	0.991675	-0.615314	0.146366
2	0.726429	-0.237653	-0.176381
3	1.044862	-0.684106	0.172282
4	1.072947	-0.666604	0.061396
5	0.918778	-0.492328	-0.043336
6	0.918002	-0.389967	-0.127984
7	0.965157	-0.519015	0.009055
8	1.036632	-0.416465	-0.133012
9	1.124920	-0.517800	-0.046829
10	1.185386	-0.533546	-0.035707
11	1.241805	-0.556381	-0.014824
12	1.038111	-0.073998	-0.276505
13	1.249574	-0.500309	0.036164
14	1.379794	-0.601899	0.048331
15	1.446032	-0.664645	0.059428
16	0.328260	0.164216	0.114883
17	1.120339	-0.229878	-0.045775
18	0.926917	-0.044050	-0.100819
19	0.820616	0.008654	-0.030265
20	0.693041	0.157544	-0.102425

THE LINEAR FILTER COEFFICIENTS  $c_1, c_2$  and  $c_3$  (FOR 3rd ORDER FILTER) FOR GAUSSIAN SAMPLES AS INPUT

Block No.	$c_1$	$c_2$	$c_3$
1	0.013515	0.010682	0.002916
2	-0.089070	-0.034122	-0.114144
3	0.080943	-0.115964	0.258790
4	0.124766	0.192269	-0.172938
5	-0.102693	0.160536	-0.154860
6	0.135906	-0.165468	-0.057347
7	-0.121415	0.092721	0.022726
8	-0.306955	-0.269164	0.095525
9	0.071145	-0.244118	-0.110240
10	0.213886	0.132074	-0.080623
11	-0.436016	-0.173864	-0.106404
12	0.113613	0.046058	-0.006265
13	0.064803	0.076313	0.047876
14	0.054312	-0.084474	0.038279
15	0.162500	-0.101040	-0.002869
16	-0.045142	0.173489	0.009722
17	-0.250265	0.069462	0.046227
18	-0.037245	-0.148705	-0.220061
19	0.061558	-0.181215	0.220685
20	-0.027743	-0.011237	0.141954

## REFERENCES

- [1] A. Papoulis, 'Probability Random Variables and Stochastic Processes', New York, McGraw-Hill, 1965.
- [2] -----, 'Maximum entropy and spectral estimation: A review', IEEE Trans. Acoust. Speech and Signal Processing, vol. ASSP-29, No.6, Dec. 1981.
- [3] J. Makhoul, 'Linear prediction: A tutorial review', Proc. IEEE, vol. 63, pp 561-580, April 1975.
- [4] J.D. Markel and A.H. Gray, Jr., 'Linear Prediction of Speech', New York, Springer-Verlag, 1976.
- [5] W.S. Dorn and D.D. McCracken, 'Numerical Methods with Fortran IV Case Studies', New York, John Wiley and Sons, 1972.
- [6] N. Levinson, 'The Wiener RMS error criterion in filter design and prediction', J. Math. Phys. vol. 25, No.4, pp 261-278, 1947.
- [7] E.A. Robinson, 'Statistical Communication and Detection', New York, Hafner, 1967.
- [8] J. Durbin, 'The fitting of time series models', Rev. Inst. Int. Statist., vol. 28, No.3, pp 233-243, 1960.
- [9] J.B. O'Neal, Jr., and R.W. Stroh, 'Differential PCM for speech and data signals', Trans. IEEE Commun. vol. COM-20, pp 900-912, Oct. 1972.

- [10] R.A. McDonald, 'Signal-to-noise and idle channel performance of differential pulse code modulation systems - particular application to voice signals', BSTJ, vol. 45, pp 1123-1151, 1966.
- [11] N.S. Jayant, 'Digital coding of speech waveforms: PCM, DPCM and DM quantizers', Proc. IEEE, vol. 62, pp 611-632, May 1974.



## MAXIMUM ENTROPY METHOD

## 3.1 STATISTICAL DEFINITION OF INFORMATION

Let a situation in which  $L$  different things  $l_i$ ,  $i = 1, \dots, L$  might happen be considered. Let the probability of occurrence be  $p_i$ . The probability of occurrence of an event is related to information. The relationship between the information and the probability is

$$I = k_1 \ln \frac{1}{p_i} \quad (3.1)$$

where  $I$  is the information and  $k_1$  a constant  $= 1$  when the logarithm is to the base 2,

If the above system is observed for a long time  $T$  ( $T$  being very large),  $p_1 T$  of  $l_1$  events,  $p_2 T$  of  $l_2$  events and so on might happen. The total information about the system is then

$$I_{\text{total}} = k_1 (p_1 T \ln \frac{1}{p_1} + p_2 T \ln \frac{1}{p_2} + \dots) \quad (3.2)$$

The average information per unit time is represented as 'H' and is referred to entropy. Thus

$$H = \frac{I_{\text{total}}}{T} = -k_1 \sum_{i=1}^L p_i \ln p_i \quad (3.3)$$

Entropy is a measure of the uncertainty described by a set of probabilities. The entropy is zero for a system where all the

$p_i$  are zero except one, which is unity. That is there is no uncertainty. Entropy is a measure of the disorder in a system.

With this knowledge of entropy, the concept of maximum entropy is fitted in for power spectral determination. The assumptions made in the conventional methods [1] concerning the data that lie outside the interval of interest, constrain the data to be either periodic or zero outside (the interval). This leads to unfortunate end effects in the spectral determination. So, what is needed, is an approach that is first of all consistent with the prior knowledge and second, estimates the prior probability assignment that describes the prior information without assuming anything beyond. This was elegantly pointed out by [2] as the first principle of data reduction, which states 'The result of any transformation imposed on the experimental data shall incorporate and be consistent with all relevant data and be maximally noncommittal with regard to unavailable data'. Jaynes [3] then, formulated his principles stating that the prior probability assignment that follows the first principle of data reduction is, the one that maximises entropy.

### 3.2 RELATIONSHIP BETWEEN MEM ESTIMATION AND LINEAR PREDICTIVE FILTERING

Most of power spectrum estimation methods depend on the knowledge of measured autocorrelation function. With the known

autocorrelation values from lags 0 to  $n$ , the  $(n+1)$ th lag value is determined as that which makes the determinant of the autocorrelation Toeplitz matrix nonnegative [4]. The entropy which relates to randomness or uncertainty is strongly connected to this correlation extension procedure. The entropy of a band limited Gaussian time series with power spectrum  $S(f)$  is proportional to

$$\int_0^W \log S(f) df \quad (3.4)$$

Under the constraint that  $S(f)$  agrees with the  $(n+1)$  measured value of the autocorrelation function, i.e.

$$\int_0^W S(f) \cos(2\pi f + \tau \delta t) df = r(\tau) ; (\tau = 0 \text{ to } n) \quad (3.5)$$

it becomes necessary to find what  $S(f)$  has the maximum entropy. Burg [5] answers this question by saying that this problem is equivalent to that of designing a  $(n+1)$  coefficients linear prediction filter with the mean square error power  $P_{N+1}$ .

The linear prediction filter theory is already dealt with in Chapter 2. The one to one correspondence between this and a AR process is also mentioned. The input to the AR model has the power spectrum as determined by equation (2.33), which is nothing but the MEM power spectrum. It is seen that the MEM process and the AR process are exactly equivalent. So, the

representation of a stochastic process by an AR model is the representation that exhibits the maximum entropy. This was indicated by [6].

### 3.3 DETERMINATION OF AUTOCORRELATION FUNCTION BY MEM

Equation (2.23) gives the recursive equation satisfied by an AR process of order M. This is

$$s(n) - c_1 s(n-1) - \dots - c_M s(n-M) = \eta(n) \quad (3.6)$$

Multiplying this equation (3.6) by  $s(n-k)$  and taking expected values, we get, for  $k > 0$

$$r(k) - c_1 r(k-1) - \dots - c_M r(k-M) = 0 \quad (3.7)$$

Since  $\epsilon(\eta(n) s(n-k)) = 0$  for  $k > 0$ .

Substituting  $k = 1$  to  $M$ , a set of equations is obtained, known as Yule-Walker equations. The Yule-Walker equations are similar to the set of equations (2.10) [7].

If  $k = 0$  is included into the equation (3.7),

$$r(0) - c_1 r(-1) - \dots - c_M r(-M) = \epsilon[\eta(n) \cdot s(n)] \quad (3.8)$$

$$= r(0) - c_1 r(1) - \dots - c_M r(M) = \sigma^2 \quad (3.9)$$

This equation (3.9) can be included into the set of equations got from equation (3.7) for  $1 \leq k \leq M$ . The final set of equations is as follows :

$$\begin{bmatrix} r(0) & r(1) & r(M) \\ r(1) & r(0) & r(M-1) \\ \vdots & \vdots & \vdots \\ r(M) & r(M-1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -c_1 \\ \vdots \\ -c_M \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (3.10)$$

The autocorrelation matrix is of size  $(M+1) \times (M+1)$ , the coefficients column vector of size  $(M+1)$ . Since, the R.H.S. column vector is also of size  $(M+1)$ ,  $\sigma^2$  can be thought of as  $P_{M+1}$ , the power output of a  $M+1$  point filter. So, equation (3.10) becomes

$$\begin{bmatrix} r(0) & r(1) & r(M) \\ \vdots & \vdots & \vdots \\ r(M) & r(M-1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ -c_M \end{bmatrix} = \begin{bmatrix} P_{M+1} \\ \vdots \\ 0 \end{bmatrix} \quad (3.11)$$

For a 2 point filter, the power output is  $P_2$  and the set of equations become

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -c_{11} \end{bmatrix} = \begin{bmatrix} P_2 \\ 0 \end{bmatrix} \quad (3.12)$$

The prediction filter can be operated equally well running backward over a data set (like the forward). This is the backward error filter already mentioned in Chapter 2. The power output of the filter can be estimated better by averaging the

outputs of the individual forward and backward error filters. The average power output of a two point forward error filter (assuming a data set of  $N$  samples), i.e. for  $M=1$

$$P_2^f = \frac{1}{N-1} \sum_{n=1}^{N-1} |s_{n+1} - c_{11}s_n|^2 \quad (3.13)$$

where  $P_2^f$  stands for the power output of a 2 point forward error filter and  $c_{11}$  is the coefficient to be determined. Similarly, the power output of a two point backward error filter is

$$P_2^b = \frac{1}{N-1} \sum_{n=1}^{N-1} |s_n - c_{11}s_{n+1}|^2 \quad (3.14)$$

This is due to the fact that the forward error filter and backward error filter are equivalent, having same coefficients. The average power of the two point filter denoted by  $P_2$  is

$$P_2 = \frac{1}{2}(P_2^f + P_2^b) \quad (3.15)$$

$$= \frac{1}{2(N-1)} \left[ \sum_{n=1}^{N-1} |s_{n+1} - c_{11}s_n|^2 + \sum_{n=1}^{N-1} |s_n - c_{11}s_{n+1}|^2 \right] \quad (3.16)$$

This power  $P_2$  is minimised with respect to  $c_{11}$  to find out  $c_{11}$ . This is done by setting the derivative of  $P_2$  with respect to  $c_{11}$  equal to zero. When this is done, the value of  $c_{11}$  is [8]

$$c_{11} = \frac{2 \sum_{n=1}^{N-1} s_n s_{n+1}}{\sum_{n=1}^{N-1} (|s_n|^2 + |s_{n+1}|^2)} \quad (3.17)$$

The value of  $r(o)$  is determined as

$$r(o) = \frac{1}{N} \sum_{n=1}^N s_n^2 \quad (3.18)$$

From equation (3.12) it is seen

$$r(1) = r(o) c_{11} \quad (3.19)$$

So, the value of  $r(1)$  can be determined. Also  $P_2$  is calculated as

$$P_2 = r(o) - r(1) c_{11} \quad (3.20)$$

Substituting equation (3.19) into equation (3.20)

$$P_2 = r(o) - r(o)c_{11}c_{11} = r(o)-r(o)c_{11}^2 \quad (3.21)$$

$$= r(o) (1-c_{11}^2) \quad (3.22)$$

It is seen that  $|c_{11}| \leq 1$

This procedure is extended to a 3 point filter with power output  $P_3$ . The set of equations is

$$\begin{bmatrix} r(o) & r(1) & r(2) \\ r(1) & r(o) & r(1) \\ r(2) & r(1) & r(o) \end{bmatrix} \begin{bmatrix} 1 \\ -c_{21} \\ -c_{22} \end{bmatrix} = \begin{bmatrix} P_3 \\ 0 \\ 0 \end{bmatrix} \quad (3.23)$$

From the second row of the matrix equation (3.23)

$$r(1) - c_{21} r(0) - c_{22} r(1) = 0 \quad (3.24)$$

From the 2 point error filter it is known that

$$r(1) = c_{11} r(0)$$

Substituting this into equation (3.24)

$$c_{11} r(0) - c_{21} r(0) - c_{22} c_{11} r(0) = 0 \quad (3.25)$$

or

$$c_{21} = c_{11} - c_{22} c_{11} \quad (3.26)$$

The corresponding power output is

$$P_3 = r(0) - r(1) c_{21} - r(2) c_{22} \quad (3.27)$$

$$= r(0) - c_{11} r(0) c_{21} - r(2) c_{22} \quad (3.28)$$

From the third row of equation (3.23),

$$r(2) = c_{21} r(1) + c_{22} r(0) \quad (3.29)$$

Substituting this value of  $r(2)$  and the value of  $c_{21}$ , from equation (3.26) into equation (3.28)

$$P_3 = P_2(1 - c_{22}^2) \quad (3.30)$$

The coefficients of the three point filter is then

$$[1, c_{11}(c_{22} - 1), -c_{22}] \quad (3.31)$$



Like in the case of two point filter, the forward error power of the three point filter is

$$P_3^f = \frac{1}{N-2} \sum_{n=1}^{N-2} |s_{n+2} - c_{21}s_{n+1} - c_{22}s_n|^2 \quad (3.32)$$

The backward error power of the three point error filter is

$$P_3^b = \frac{1}{N-2} \sum_{n=1}^{N-2} |s_n - c_{21}s_{n+1} - c_{22}s_{n+2}|^2 \quad (3.33)$$

The overall average power  $P_3$  is then

$$P_3 = \frac{1}{2} (P_3^f + P_3^b) \quad (3.34)$$

The only unknown is  $c_{22}$ . Differentiating equation (3.34) with respect to  $c_{22}$  and equating to zero yields the value of  $c_{22}$ . It is seen that the value of  $c_{22}$  must not exceed unity.

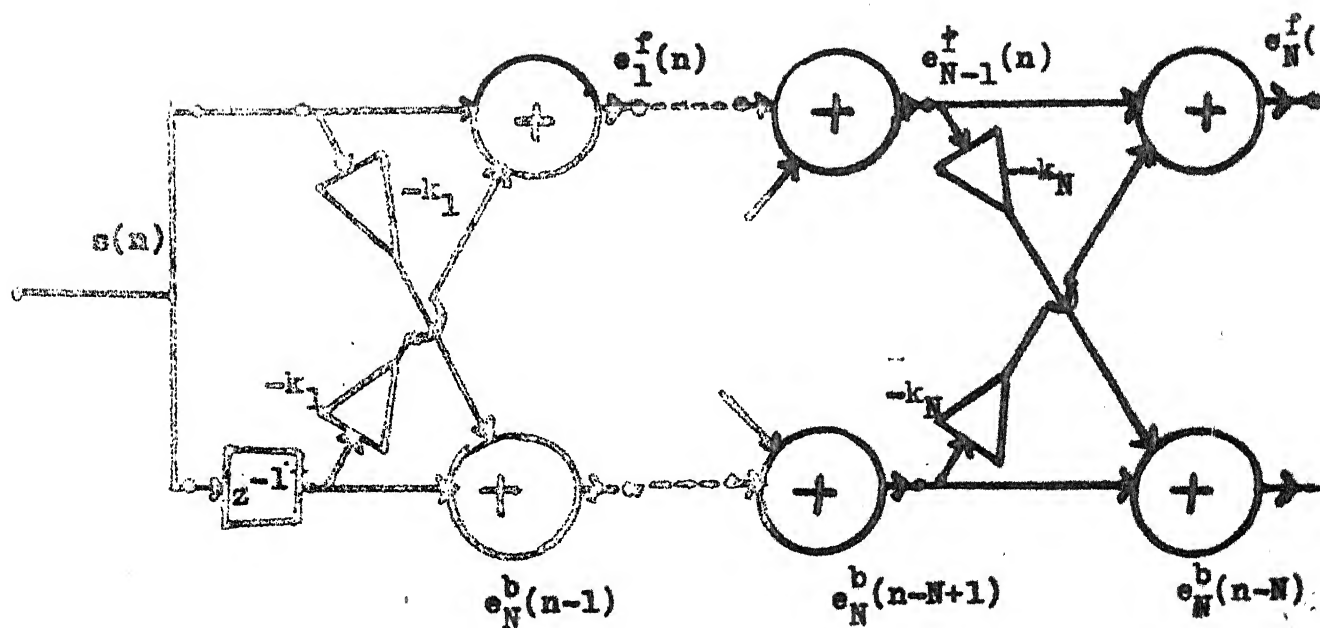
The procedure can be extended to a four point filter and so on. The general equation for the filter coefficient  $c_{mk}$  for  $1 \leq m \leq M$  and  $1 \leq k \leq M-1$ , is

$$c_{mk} = c_{m-1,k} - c_{mm} \cdot c_{m-1,m-k} \quad (3.35)$$

and

$$P_m = P_{m-1}(1 - c_{mm}^2) \quad (3.36)$$

The unknown autocorrelation values are determined thus by recursively calculating the coefficients of the filter. The recursive equation for the autocorrelation coefficient is given as



**Fig 3.1 Lattice Filter**

$$r(m) = \sum_{n=1}^k r(m-n) c_{mn} \quad (3.37)$$

### 3.4 STABILITY

The combination of a forward error filter and backward error filter is shown in Fig. 3.1. This overall system is also generally known as Lattice filter. For example, in the figure is shown  $N$  sections in cascade. Each section has one delay element and two multipliers. The coefficients  $c_{k-1,k}$  are nothing but the reflection or PARCOR coefficients of the lattice. It is seen that these coefficients have magnitude  $\leq 1$ , which is a necessary and sufficient condition for the filter to be a prediction error filter. By checking the absolute value of these reflection coefficients to be  $< 1$  at every recursion, stability is assured. This is a consequence of Durbin's modification of Levinson's algorithm, as outlined in Chapter 2.

The forward error filter has system function

$$H_N^f(z) = 1 - \sum_{k=1}^N a_k^N z^{-k} \quad (3.38)$$

for this filter to be stable this polynomial must be Hurwitz i.e., all its roots  $z_i$  lie inside the unit circle. Correspondingly, it follows that all roots of the backward error filter with

$$H_N^b(z) = H_N^f(1/z) \quad (3.39)$$

lie outside the unit circle. A. Papoulis [9] gives a proof by induction to show that the filters are stable.

### 3.5 ORDER OF MEM PREDICTION FILTER

Proper choice of the order of the filter or AR process is an important part of the MEM analysis. Various criteria have been suggested to determine the same [10]. It has been noticed by [7] that spurious details were introduced into the spectrum by excessively large values of order. Small values tend to smoothen out the spectrum thus annihilating some significant details. However, the final prediction error (FPE) criterion of Akaike [11] is the most widely used. The  $c_{mk}$ 's are a set of AR coefficients based on a certain sample realisation of a time series. Akaike's FPE measures the error that is incurred on the average when these coefficients are used as the model for other realization of the same time series. For zero mean processes, the FPE is given by

$$FPE_m = \frac{N+(m+1)}{N-(m+1)} P_m \quad (3.40)$$

The order of the AR model is chosen as one that minimises the FPE. It has been observed to give a good results if order of the process is kept limited to  $N/2$ .

THE NORMALIZED AUTOCORRELATION COEFFICIENTS OF SPEECH SAMPLES  
CALCULATED BY MEM (FOR BLOCKS OF 50 SAMPLES EACH)

Block No.	r(0)	r(1)	r(2)	r(3)
1	1.0	0.622342	0.092937	-0.144060
2	1.0	0.575682	0.079000	-0.255806
3	1.0	0.628727	0.081145	-0.173048
4	1.0	0.646241	0.066455	-0.298088
5	1.0	0.614346	0.045496	-0.303995
6	1.0	0.649136	0.122861	-0.268339
7	1.0	0.635983	0.100567	-0.223966
8	1.0	0.710651	0.225693	-0.195012
9	1.0	0.732754	0.272176	-0.120074
10	1.0	0.764918	0.345863	-0.033843
11	1.0	0.793901	0.417720	0.062192
12	1.0	0.824057	0.553609	0.237223
13	1.0	0.847070	0.588801	0.348118
14	1.0	0.881156	0.656502	0.423801
15	1.0	0.892936	0.679633	0.448714
16	1.0	0.442270	0.360205	0.305752
17	1.0	0.884131	0.720177	0.557825
18	1.0	0.826158	0.638436	0.454566
19	1.0	0.808019	0.647273	0.507890
20	1.0	0.749619	0.600230	0.431588

THE NORMALIZED AUTOCORRELATION COEFFICIENTS OF GAUSSIANS  
SAMPLES CALCULATED BY MEM (FOR BLOCKS OF 50 SAMPLES EACH)

Block No.	r(0)	r(1)	r(2)	r(3)
1	1.0	0.013693	0.010907	0.003210
2	1.0	-0.084264	-0.016998	-0.109755
3	1.0	0.049543	-0.099133	0.245021
4	1.0	0.114480	0.186754	-0.127627
5	1.0	-0.159526	0.201623	-0.201175
6	1.0	0.124272	-0.155705	-0.099071
7	1.0	-0.131177	0.105667	-0.002266
8	1.0	-0.258009	-0.214613	0.230848
9	1.0	0.079090	-0.247210	-0.147135
10	1.0	0.231302	0.162898	-0.015232
11	1.0	-0.374069	0.029038	-0.054028
12	1.0	0.118712	0.058801	0.005883
13	1.0	0.074548	0.084713	0.059054
14	1.0	0.047254	-0.080098	0.029937
15	1.0	0.147789	-0.077449	-0.030387
16	1.0	-0.052555	0.175350	-0.007311
17	1.0	-0.262832	0.123090	-0.002835
18	1.0	-0.004139	-0.147640	-0.213946
19	1.0	0.019275	-0.175775	0.206372
20	1.0	-0.029485	-0.014604	0.142691

## REFERENCES

- [1] B.V.K. Vijaykumar, 'Power spectrum using maximum entropy method', M.Tech. Thesis, IIT Kanpur, 1977.
- [2] J.G. Ables, 'Maximum entropy spectral analysis', Astron. Astrophys. Suppl. Series, vol. 15, pp 383-393, 1974.
- [3] E.T. Jaynes, 'Prior probabilities', IEEE Trans. Systems Sci. Cybern, vol. SEC 4, pp 227-241, Sept. 1968.
- [4] J.P. Burg, 'Maximum entropy spectral analysis', Paper presented at the 37th meeting of SEG, Oklahomacity, Oct. 1967.
- [5] -----, 'A new analysis technique for time series data', Paper presented tat the NATO advanced study institute on signal processing , Netherlands, Aug. 1968.
- [6] A. Van Den Bos, 'Alternative interpretation of maximum entropy spectral analysis', IEEE Trans. Inform. Th. vol. IT-17, pp 493-494, July 1971.
- [7] T.J. Ulrych and T.N. Bishop, 'Maximum entropy spectral analysis and autoregressive decomposition', Rev. Geophysics and Space Phys. vol. 13, pp 183-200, Feb. 1975.
- [8] R.A. Monzingo and T.W. Miller, 'Introduction to adaptive arrays', New York, John Wiley and Sons, 1980.

- [9] A. Papoulis, 'Maximum entropy and spectral estimation : A review,' IEEE Trans. Acous. Speech and Signal Processing, vol. ASSP-29, No.6, Dec. 1981.
- [10] H. Akaike, 'Fitting autoregressive models for prediction', Ann. Inst. Statist. Math. vol. 21, pp 243-247, 1969.
- [11] -----, 'Power spectrum estimation through autoregressive model fitting', Ann. Inst. Statist. Math., vol. 21, pp 407-419, 1969.



## CHAPTER 4

### SYSTEM DESCRIPTION

#### 4.1 OPTIMUM QUANTIZATION

The purpose of a quantizer is to map the input within a given range to an output level. This enables a continuous range of input to be restricted to a certain number of output levels, for onward digital transmission. The quantizer is specified by the end points  $x_k$  of the ' $N_1$ ' input ranges and an output level  $y_k$  corresponding to each input range. Based on some distortion measure, the  $y_k$ 's for a given  $N_1$  are chosen along with the associated  $x_k$ 's. Max[1] uses a distortion measure 'D' as the expected value of  $f(e)$  where  $f(\cdot)$  is some function and  $e$  is the quantization error. If the input amplitude probability density is  $p(x)$  then

$$D = \epsilon[f(\text{Quantizer In} - \text{Quantizer out})] \quad (4.1)$$

$$= \sum_{i=1}^{N_1} \int_{x_i}^{x_{i+1}} f(x-y_i) p(x) dx \quad (4.2)$$

where  $x_{N_1+1} = \infty$  and  $x_1 = -\infty$  and for an input  $x$  between  $x_i$  and  $x_{i+1}$ , the output of the quantizer is  $y_i$ . To minimise  $D$ , equation (4.2) is partially differentiated with respect to  $x_i$ 's and  $y_i$ 's and setting them equal to zero. This yields as  $p(x) \neq 0$ .

$$f(x_k - y_{k-1}) = f(x_k - y_k); k = 2, \dots, N_1 \quad (4.3)$$

and

$$\int_{x_j}^{x_{j+1}} f'(x-y_j) p(x) dx = 0, \quad j = 1, 2, \dots, N_1 \quad (4.4)$$

The function  $f(x)$  to be used must be a good metric, i.e.,  $f(x)$  must be monotonically nondecreasing, i.e.,

$$f(0) = 0 \quad (4.5)$$

and

$$f(x) = f(-x) \quad (4.6)$$

If a cost function  $f(e)$  proportional to square of error is assumed as

$$f(e) = e^2 \quad (4.7)$$

Equation (4.3) becomes

$$(x_k - y_{k-1})^2 - (x_k - y_k)^2 = 0 \quad (4.8)$$

or

$$x_k = (y_k + y_{k-1})/2, \quad k = 2, \dots, N_1 \quad (4.9)$$

Equation (4.4) becomes

$$\int_{x_j}^{x_{j+1}} (x-y_j) p(x) dx = 0; \quad j = 1, 2, \dots, N_1 \quad (4.10)$$

That is  $y_j$  is the centroid of the area of  $p(x)$  between  $x_j$  and  $x_{j+1}$ .

The above two equations (4.9) and (4.10) are solved using numerical methods as they are not readily amenable to any

standard analytical methods. Assuming  $p(x)$  to be Gaussian for different  $N_1$ , Max [1] obtained solutions numerically and gave the  $x_i$ 's and  $y_i$ 's in a tabular form. It is seen that the criterion used for distortion minimisation is MMSE.

Speech, although has been approximated as Gaussian by various researchers, is not strictly Gaussian. Davenport [2] and Richards [3] have made extensive studies with a large number of speakers to arrive at a more accurate model for the probability density of speech. Following their results, McDonald [4] proposed a special form of gamma density as

$$p(x) = \left[ \frac{\sqrt{3}}{8\pi\sigma_x} \right]^{\frac{1}{2}} \exp\left[-\frac{\sqrt{3} x}{2\sigma_x}\right] \quad (4.11)$$

where  $\sigma_x^2$  is the variance of input  $x$ . A slightly poorer approximation is the Laplacian density given as

$$p(x) = \frac{1}{\sqrt{2} \sigma_x} \exp\left[-\frac{\sqrt{2} |x|}{\sigma_x}\right] \quad (4.12)$$

It is reported that the above two densities serve as very good approximations to real speech density.

Following exactly the procedure of Max, Paez and Glisson [5] assumed  $p(x)$  to be Laplacian and Gamma and solved for  $x_i$ 's and  $y_i$ 's. Table 4.1 shows the values for  $N_1 = 2, 4, 8$  and 16 (i.e.) for binary 1, 2, 3 and 4 bit quantizers. The 'minimum distortion' factor of Max was called as 'optimum' by them,

Table 4.1

Optimum quantizer for Laplacian density

$N_1$	2		4		8		16	
$i$	$y_i$	$y_i$	$x_i$	$y_i$	$x_i$	$y_i$	$x_i$	$y_i$
1	$\infty$	0.707	1.102	0.395	0.504	0.222	0.266	0.126
2			$\infty$	1.810	1.181	0.785	0.566	0.407
3					2.285	1.576	0.910	0.726
4					$\infty$	2.994	1.317	1.095
5							1.821	1.540
6							2.499	2.103
7							3.605	2.895
8							$\infty$	4.316
9								
10								

## 4.2 GENERAL DIFFERENTIAL QUANTIZER-PREDICTOR

Speech signal inherently has high correlation between adjacent samples. So, the variance of the first difference is smaller than the variance of the input itself. That is,

$$\langle e_n^2(1) \rangle = \langle (s_n - s_{n-1})^2 \rangle \quad (4.13)$$

$$= 2 \langle s_n^2 \rangle (1 - r(1)) \quad (4.14)$$

where  $e_n(1)$  is the first difference and  $n$  denotes that it corresponds to the  $n$ th sample;  $r(1)$  is the autocorrelation between the adjacent samples. If  $r(1)$  is  $> 0.5$ , obviously the difference or error has smaller variance. For a given fineness of quantization, the quantization error power is proportional to the variance of the quantizer input. So, by allowing the error to be the input to the quantizer rather than the actual input itself, a better SNR could be obtained. Or equivalently for same SNR, fewer number of bits are enough for quantization of the signal. Generalizing equation (4.13) by subtracting a weighted (by  $c_1$ ) times of the previous sample,

$$\langle e_n^2(1) \rangle = \langle (s_n - c_1 s_{n-1})^2 \rangle \quad (4.15)$$

$$= \langle s_n^2 \rangle (1 + c_1^2 - 2c_1 r(1)) \quad (4.16)$$

The above equation has a minimum when  $c_1 = r(1)$ . A further generalization is done by subtracting  $\sum_{k=1}^M c_k s_{n-k}$  from  $s_n$

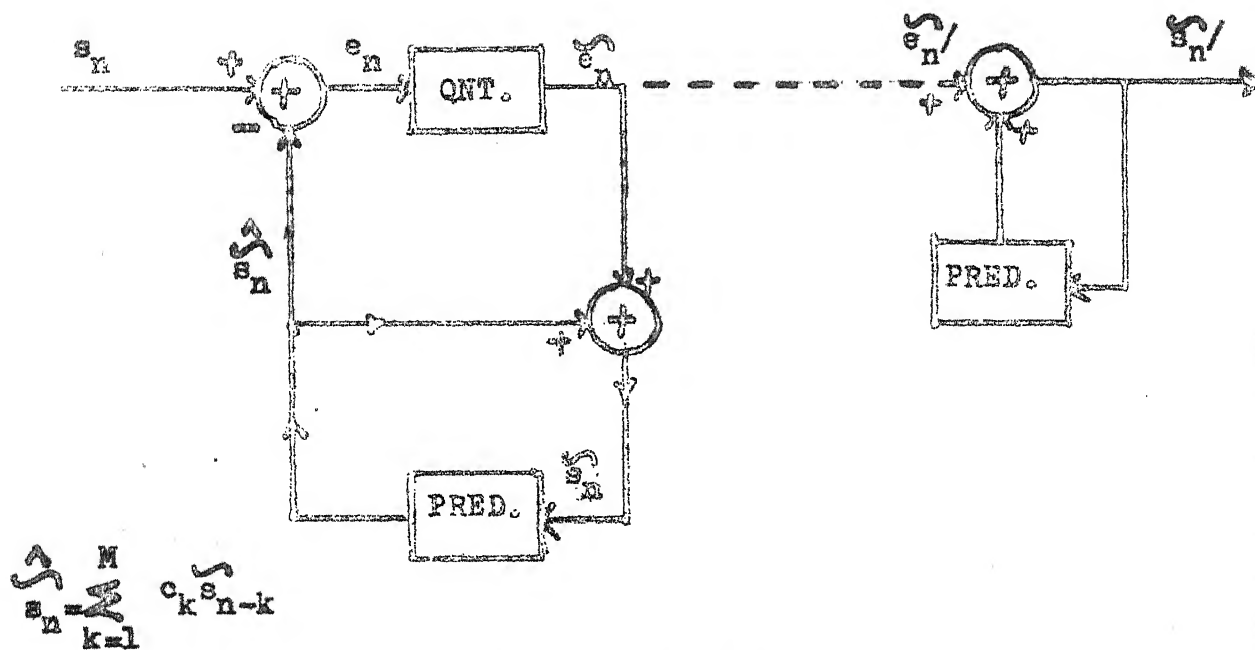


Fig.4.1 General Block diagram of DPCM coder-decoder

and this becomes the quantizer input. This is the basic principle of differential coding [6].

A general block diagram of DPCM coder-decoder is shown in Fig. 4.1. Here  $e_n$ , the differences or error signal is the quantizer input, producing  $\tilde{e}_n$  as the output of quantizer. The prediction is based on  $M$  past quantized values. The coefficients of the predictor are  $c_k$ ,  $k = 1, \dots, M$ . From the figure 4.1, the quantization error for the  $n$ th sample is

$$E_n = e_n - \tilde{e}_n = (s_n - \hat{s}_n - \tilde{e}_n) \quad (4.17)$$

$$\hat{s}_n = (\hat{s}_n + \tilde{e}_n) \quad (4.18)$$

So,

$$E_n = (s_n - \hat{s}_n) \quad (4.19)$$

The predicted value is  $\hat{s}_n$  and it is

$$\hat{s}_n = \sum_{k=1}^M c_k \tilde{s}_{n-k} = \sum_{k=1}^M c_k (s_{n-k} - E_{n-k}) \quad (4.20)$$

Overall SNR (signal to noise power ratio) is

$$\text{SNR} = \langle s_n^2 \rangle / \langle E_n^2 \rangle = \frac{\langle s_n^2 \rangle}{\langle (s_n - \hat{s}_n)^2 \rangle} \quad (4.21)$$

The gain  $G$  over PCM is the ratio of the variance of  $s_n$  to that of  $e_n$

$$\text{Gain} = G = \langle s_n^2 \rangle / \langle e_n^2 \rangle = \frac{\langle s_n^2 \rangle}{\langle (s_n - \hat{s}_n)^2 \rangle} \quad (4.22)$$

To maximize gain the predictor coefficients  $c_k$ 's are chosen to minimise the denominator of equation (4.22). With MMSE as criterion, this is done as mentioned in Chapter 2.

### 4.3 PERFORMANCE MEASURES

It is well known in speech analysis that the SNR is not a perfect indicator of speech quality and intelligibility. This is because perceived quality and intelligibility depend on the subjective loudness of the quantization noise and not just on the quantization noise power. Due to the auditory masking properties of the human ear, the subjective loudness is determined by the spectrum of the quantization noise and its relation to the input signal spectrum [7],[8], [9]. The SNR does not reflect these considerations. Still, it is the most popular performance measure.

It is common to do the time averaging in eqn. (4.21) over the entire period of the utterance or sentence processed. It is not necessary to do so; one can use shorter averaging intervals. A recently proposed approach is to compute the SNR over many nonoverlapping blocks of data within an utterance [10]. The time variation of the resulting short term SNR provides an indication of how well the coder is performing over various blocks of data. Based on these short term SNR values denoted by  $\text{SNR } B_j(\text{dB})$ , ( $j$ th block SNR in dB) a performance measure SNRSEG (segmental SNR) is defined as



$$\text{SNRSEG} = \frac{1}{k} \sum_{j=1}^k \text{SNR } B_j (\text{dB}) \quad (4.23)$$

Infrequent large values of  $\text{SNR } B_j$  tend to show up better in SNRSEG than in SNR [11]. The short term  $\text{SNR } B_j$  is a useful performance measure for coders with adaptation in quantizer and predictor in a block fashion.

The gain of equation (4.22) is also called as SPER in predictive systems like DPCM. That is SPER is motivated by a desire to evaluate the predictor performance of DPCM and due to the closed loop nature of DPCM with the presence of quantizer within the loop, SPER is also affected by the quantizer.

Another performance indicator for predictive systems is SNRI defined as [12], [13]

$$\text{SNRI} = \frac{\langle s_n^2 \rangle}{\langle (s_n - \sum_{k=1}^M c_k s_{n-k})^2 \rangle} \quad (4.24)$$

where the prediction is based on the past inputs rather than on the past quantizer outputs like in DPCM. SNRI is interpreted as the amount by which linear prediction can reduce the input signal power and hence sometimes used as a measure of maximum utility of linear prediction. It is seen SPER and SNRI are closely related. Since SPER is computed based on the actual quantizer input  $e_n$ , it is the true ratio of signal power

to prediction error power in DPCM. However, SNRI upper bounds SPER and hence SNRI is useful as an indicator of the maximum performance improvement available.

Interesting relationships between SNR, SPER and SNRI have been derived by Gibson [12].

#### 4.4 ADAPTIVE QUANTIZATION

The quantizer design has to be compromised between two contradictory requirements. To accommodate a high peak to peak signal a wide step size is preferred. At the same time to reduce quantization distortion, it is necessary to have a smaller step size. Speech depicts a wide fluctuation in the instantaneous amplitude level. The amplitude variations depend on the speaker, the environment and has a wide swing between voiced and unvoiced segments in a given utterance.

To accommodate these amplitude fluctuations, very often the properties of the quantizer are made adaptive. There are two basic approaches. In the first, means are provided to match the step size of the quantizer to the signal variance at the quantizer input. In the second method, the quantizer remains fixed with no alteration to step size, but is preceded by an amplifier with a time varying gain. The time varying gain is calculated based on an estimate of the short term variance of the block of the signal under consideration. Both the methods are equivalent in operation [13].

The quantizer adaptation can be either feed forward or backward as the case may be. When the adaptation is effected from parameters calculated directly from the input signal it is known as feed forward adaptation. When the adaptation parameters are determined based on the output of the quantizer it is backward adaptation.

Noll [10] gives an exhaustive account of the adaptation with a variable gain amplifier in front of the quantizer. The scheme is depicted in Fig. A.1. The adaptation parameter is to be proportional to the variance of the input to the quantizer. However, since the error signal  $e(n)$  is proportional to the input, it is reasonable to calculate the adaptation parameters from the input signal [14]. The short term variance is defined as

$$\sigma^2(k) = \frac{1}{L} \sum_{n=1}^L s^2(n) \quad (4.25)$$

where  $L$  is the number of samples in the block ' $k$ '. With the knowledge of  $\sigma^2(k)$ , the gain for the  $k$ th block of the speech segment is [14] ( $G$  being the gain),

$$G(k) = \frac{1}{\sigma(k)} \quad (4.26)$$

That is the gain of the amplifier is inversely proportional to the standard deviation of the block under consideration. This necessitates the transmission of the adaptation parameter ( $\sigma$ ) to the receiver for every block. This transmission would account

for a marginal increase in data rate. In the case of backward adaptation scheme, no transmission separately of the adaptation parameter is necessary. But, it suffers from the disadvantage of increased sensitivity to channel errors, since from the quantizer output, not only the signal is reconstructed at the receiver, but also the adaptation parameter is extracted. So channel errors cause an error not only in the quantizer level but also in the quantizer step size.

The method of step size adaptation, either feed forward or backward, is also equally popular. Various algorithms have been reported [15], [16]. Most of these methods modify the step size (for every new input sample) by a factor depending on the knowledge of which quantizer slots were occupied by the previous samples. In a simple form the scheme operates with a one word memory. The output of a uniform quantizer of  $N_1$  bits is written as

$$y_n = H_n \frac{\delta_n}{2} \quad (4.27)$$

where  $y_n$  is the output for the  $n$ th sample :

$$\pm H_n = 1, 3, 5 \dots 2^{N_1-1} \text{ and}$$

$\delta_n$  is the step size for that sample.

Now the step size  $\delta_{n+1}$  is chosen to be the previous step size multiplied by a time invariant function  $M$  of the code word magnitude  $H_n$ , (i.e.)

$$\delta_{n+1} = \delta_n \cdot M(H_n) \quad (4.28)$$

where the multiplier function  $M$  is properly designed, the adaptation method serves to match the step size, at every sample, to an updated estimate of the signal variance.

Depending on the update interval of the adaptation, it is referred to as either instantaneous or syllabic (approximately equal to the duration of a syllable utterance). The scheme governed by equations (4.27) and (4.28) is instantaneous. Whereas the variable gain adaptation scheme is syllabic with the update time varying from 4 ms to nearly 16 ms.

Employing adaptation of quantizer results in a better matching of the quantizer input to the short term signal variance. This reflects as an increase in the SNR performance. Approximately 4 - 6 dB improvement has been reported in the literature.

#### 4.5 ADAPTIVE PREDICTION

The spectral properties of speech vary widely from speaker to speaker and also within a speech segment especially between voiced and unvoiced portions. Due to the nonstationary nature of speech, a fixed predictor cannot predict the signal efficiently at all times [17]. For example, speech is approximately periodic during voiced portions. Then a good prediction is possible based on the value of the signal one period earlier. However, the period of speech varies with time. So, the

predictor coefficients must change with the changing period of the input signal. This is accomplished by making the predictor adaptive. The number of articles available in the literature on this topic seems to be limited. Two reasons for this are : (i) prediction adaptation is generally felt to be too complex and (ii) the performance improvement due to adaptive prediction is not substantial [12]. McDonald's widely quoted theoretical results says that 60 percent of the available improvement from the predictor is obtained with a first order predictor [4]. But to obtain good quality of speech at data rates less than 16 Kb/s predictor performance is to be improved by making it adaptive. The most popular method of adaptation of the predictor is to update the autocorrelation coefficients and hence the predictor coefficients over durations of 4 ms to 16 ms. During this duration local stationarity is assumed.

A complicated predictor adaptation algorithm is presented in [17]. The overall predictor structure is a combination of the two stage predictors - one (PR1) to remove periodicity due to pitch or long term redundancy, and another (PR2) to remove the short term redundancy due to vocal tract characteristics and the shape of the glottal wave. The two predictors in z-transform notation are represented as follows :

$$PR1(z) = \gamma z^{-M_1} \quad (4.29)$$

$$PR2(z) = \sum_{i=1}^{M_2} c_i z^{-i} \quad (4.30)$$

where  $\gamma$  and  $M_1$  are determined later.

The overall  $PR(z)$  of the entire predictor is

$$PR(z) = [PR1(z) + PR2(z) [1-PR1(z)]] \quad (4.31)$$

$$= [\gamma z^{-M_1} + \sum_{i=1}^{M_2} c_i z^{-i} (1-\gamma z^{-M_1})] \quad (4.32)$$

The predictor  $PR2(z)$  is obtained in the normal way like any other linear predictor. The parameters  $\gamma$  and  $M_1$  are obtained as follows :

$M_1$  is selected to maximize the normalized correlation

$$r_{M_1} = \frac{\sum_{n=1}^k s_n s_{n-M_1}}{[\sum_{n=1}^k s_n^2 \sum_{m=1}^k s_{m-M_1}^2]^{\frac{1}{2}}} \quad (4.33)$$

The optimum value of  $M_1(\text{opt})$  is found by a search of computed and tabulated values of  $r_{M_1}$ .  $\gamma$  is set to have the value

$$\gamma_{\text{opt}} = \frac{\sum_{n=1}^k s_n s_{n-M_1(\text{opt})}}{\sum_{n=1}^k s_{n-M_1(\text{opt})}^2} \quad (4.34)$$

Typically  $k = 75$  and  $M_1(\text{max})$  is 150. Though performance wise this scheme is very good, due to the large amount of computations, adaptive predictors of this type are too complex to most real time applications.

Like in adaptive quantization, feed forward and backward adaptive prediction is possible. One of the backward adaptive schemes is the gradient method [18]. It is reported by Cummiskey [18] that his scheme, a combination of backward adaptive quantization and backward adaptive prediction failed miserably. According to him, the quantizer and the predictor are significant contributors to this failure. There are also later studies which report that the backward adaptive scheme is seriously disturbed by transmission errors of the channel[19].

#### 4.6 TRANSMISSION OF PARAMETERS

The linear predictor coefficients and the quantizer adaptation parameter are conveyed to the receiver once in every update interval. Earlier methods directly quantized these parameters and sent to the receiver. Itakura's scheme [20] coded each parameter by 9 bits. It is noted that a relatively high accuracy is needed (8-10 bits per coefficient) [21]. Typically, for a first order predictor with 9 bits coding per coefficient, 18 bits are needed for transmission of these parameters during every update duration. If this duration is nearly 20 ms, then 900 bits are needed per sec for conveying of parameters to the distant receiver. This adds up to the data rate necessary for transmission of the quantized differences to give the overall data rate. For three predictor coefficients 1800 bits are required for transmission of parameters.



Instead of quantizing the parameter directly and sending them, a better choice would be to use the reflection coefficients, (though this does not give any reduction in the bandwidth necessary for sending these). This is because, positive definiteness of the autocorrelation coefficients is not always ensured when the parameters are quantized [22]. This leads to the instability of the prediction filter. Also, the quality of the decoded speech is a function of the 'maximum perceptual error' between the decoded and the original speech. A reasonable criterion is therefore to minimize the maximum perceptual error. It is assumed that an accurate representation of the power spectrum is necessary for good quality [22]. Thus, the criterion for optimal quantization of parameters, is to minimize the maximum spectral error due to quantization. This is done by transforming the reflection coefficients to the log area\* ratios and linearly quantizing them.

Recently, it is proposed that two kinds of transformations, inverse sine and inverse hyperbolic tangent can be used [23], to be followed by a uniform quantizer. It is also pointed out that the precision with which each reflection coefficient is to be encoded varies, the higher order coefficients requiring less precision. The bit allocations for each parameters and the maximum to minimum variation necessary for encoding the reflection coefficients are given in [23].

---

\*The term originates from the relation of glottal area to the reflection coefficients in the modelling to obtain the vocal tract area function estimates.

## BLOCK SNR FIGURES FOR SPEECH SAMPLES

BLOCK SIZE - 50 ; QUANTIZER - 3 bit, adaptive;

PREDICTOR - First order adaptive

---

Blocks No.	Block SNR (dB)
1	13.124343
2	13.045203
3	15.637971
4	16.261671
5	15.204613
6	14.124889
7	16.064677
8	16.582332
9	16.956900
10	15.343028
11	14.359510
12	16.524025
13	16.900963
14	16.023976
15	16.694060
16	9.023916
17	16.683720
18	16.643762
19	16.515311
20	16.178700

---

Segment SNR is 15.394679 dB

## BLOCK SNR FIGURES FOR SPEECH SAMPLES

BLOCK SIZE - 50; QUANTIZER - 3 bit adaptive

PREDICTOR - Second order adaptive

---

Block No.	Block SNR (dB)
1	15.684679
2	13.711535
3	15.377128
4	16.902858
5	16.570192
6	15.625226
7	15.514239
8	16.268007
9	16.493048
10	15.972446
11	16.756858
12	17.084521
13	17.489089
14	17.120869
15	16.635226
16	8.932692
17	16.053062
18	17.011274
19	16.522627
20	16.999514

---

Segment SNR is 15.936255 dB

## BLOCK SNR FIGURES FOR SPEECH SAMPLES

BLOCK SIZE - 50; QUANTIZER - 3 bit, adaptive

PREDICTOR - Third order, adaptive

Block No.	Block SNR (dB)
1	16.644759
2	13.969165
3	17.372970
4	16.878557
5	16.144523
6	16.382229
7	15.597213
8	17.487268
9	16.475826
10	16.229284
11	17.262019
12	16.640277
13	17.115826
14	17.131104
15	16.899681
16	8.812526
17	15.784485
18	18.177828
19	15.915454
20	16.038848

Segment SNR is 16.147992 dB

## BLOCK SNR FIGURE FOR SPEECH SAMPLES

BLOCK SIZE - 50; QUANTIZER - 4 bit adaptive

PREDICTOR - First order adaptive

 $\delta$ 

Block No.	Block SNR (dB)
1	22.759317
2	21.333798
3	19.596347
4	20.105386
5	18.534467
6	18.418346
7	18.419426
8	21.740647
9	20.878594
10	20.500616
11	20.662686
12	21.966126
13	21.015103
14	22.719909
15	22.561036
16	14.860628
17	22.668761
18	22.254577
19	21.234538
20	21.273964

Segment SNR is 20.675214 dB

## BLOCK SNR FIGURES FOR SPEECH SAMPLES

BLOCK SIZE - 50; QUANTIZER - 4 bit, adaptive

PREDICTOR - Second order, adaptive

Block No.	Block SNR (dB)
1	18.38610
2	20.59669
3	20.96681
4	21.16609
5	20.33772
6	21.77319
7	21.39272
8	22.04423
9	22.00732
10	21.72962
11	21.58912
12	20.45459
13	20.62245
14	23.25254
15	22.13913
16	16.06377
17	20.97418
18	22.22658
19	21.08975
20	21.85206

Segment SNR is 21.03368 dB

## BLOCK SNR FIGURES FOR SPEECH SAMPLES

BLOCK SIZE - 50; QUANTIZER - 4 bit adaptive

PREDICTOR - Third order adaptive

Block No.	Block SNR (dB)
1	18.80518
2	22.95931
3	20.55761
4	20.43231
5	19.87662
6	19.97059
7	21.47560
8	21.32536
9	21.92312
10	23.19771
11	21.70337
12	21.12495
13	21.31313
14	21.73176
15	22.76065
16	15.57234
17	21.27013
18	21.34159
19	21.87370
20	22.22883

Segment SNR is 21.072198

## BLOCK SNR FIGURES FOR SPEECH SAMPLES

BLOCK SIZE 100; QUANTIZER - 4 bit adaptive

PREDICTOR - Third order adaptive

Block No.	Block SNR (dB)
1	20.787213
2	22.028668
3	20.841824
4	21.927631
5	21.439780
6	22.239482
7	21.769214
8	13.081285
9	21.969623
10	21.694146

Segment SNR is 20.777887 dB



BLOCK SNR FIGURES FOR GAUSSIAN SAMPLES  
BLOCK SIZE - 50; QUANTIZER - 3 BIT, ADAPTIVE  
PREDICTOR - First order adaptive

Block No.	Block SNR (dB)
1	14.176498
2	15.041225
3	14.441862
4	13.569500
5	12.963721
6	14.192600
7	14.074719
8	14.454479
9	14.964948
10	14.876275
11	15.050324
12	13.718763
13	13.948370
14	13.789433
15	12.059241
16	12.719600
17	14.115815
18	14.060949
19	15.278959
20	12.887230

Segment SNR is 14.019225 dB

## BLOCK SNR FIGURES FOR GAUSSIAN SAMPLES

BLOCK SIZE - 50; QUANTIZER - 3 bit adaptive

PREDICTOR - Second order adaptive

Block No.	Block SNR (dB)
1	14.092083
2	14.627620
3	14.700254
4	14.807927
5	13.470818
6	14.305550
7	14.073708
8	15.526875
9	14.251805
10	14.971290
11	14.448292
12	13.839994
13	15.113131
14	13.457106
15	13.200611
16	13.737909
17	13.874761
18	14.120780
19	13.696516
20	12.939654

Segment SNR is 14.162834 dB

## BLOCK SNR FIGURES FOR GAUSSIAN SAMPLES

BLOCK SIZE - 50; QUANTIZER - 3 bit, adaptive

PREDICTOR - Third order adaptive

Block No.	Block SNR (dB)
1	14.095141
2	14.994264
3	13.975322
4	15.685120
5	13.871730
6	14.465337
7	13.895567
8	15.409681
9	14.249836
10	15.726452
11	14.046402
12	13.837407
13	15.354870
14	13.581699
15	13.222607
16	13.707962
17	14.215344
18	14.234654
19	13.898327
20	13.999783

Segment SNR is 14.323375 dB

## BLOCK SNR FIGURES FOR GAUSSIAN SAMPLES

BLOCK SIZE - 50 ; QUANTIZER - 4 bit adaptive

PREDICTOR - First order adaptive

---

Block No.	Block SNR (dB)
1	19.590659
2	19.850695
3	20.577953
4	19.471216
5	18.492211
6	18.433614
7	19.997118
8	19.438855
9	18.360900
10	19.429696
11	20.191251
12	18.958665
13	19.837039
14	19.241767
15	19.552083
16	19.526083
17	18.563282
18	20.295842
19	18.675315
20	19.344851

---

Segment SNR is 19.391456 dB

## BLOCK SNR FIGURES FOR GAUSSIANS SAMPLES

BLOCK SIZE - 50; QUANTIZER - 4 bit adaptive

PREDICTOR - Second order adaptive

---

Block No.	Block SNR (dB)
1	19.589179
2	19.256260
3	19.050061
4	19.943994
5	19.894579
6	18.800029
7	20.022684
8	18.990258
9	18.384662
10	19.333676
11	20.417503
12	19.058788
13	19.968661
14	19.031781
15	19.740462
16	19.571316
17	18.727680
18	19.674602
19	19.035638
20	19.861923

---

Segment SNR is 19.417687 dB

---

## BLOCK SNR FIGURES FOR GAUSSIAN SAMPLES

Block size - 50; Quantizer - 4 bit adaptive

Predictor - Third order, adaptive

Block No.	Block SNR (dB)
1	19.466603
2	19.140464
3	18.220989
4	20.079584
5	19.384422
6	19.630827
7	20.697524
8	20.411218
9	17.702670
10	19.637062
11	20.307969
12	19.498857
13	19.443954
14	18.249659
15	20.015668
16	18.785969
17	18.990143
18	20.148510
19	20.783491
20	19.592672

Segment SNR is 19.509413 dB

## REFERENCES

- [1] J. Max, 'Quantizing for minimum distortion', IRE Trans. Inform. Theory, vol. IT-6, pp 7-12, March 1960.
- [2] W.B. Davenport, 'An experimental study of speech wave probability distributions', J. Acoust. Soc. Am., vol. 24, pp 390-399, July 1952.
- [3] D.L. Richards, 'Statistical properties of speech signals', Proc. IEE, 111, No.5, May 1964.
- [4] R.A. MacDonald, 'Signal to noise and idle channel performance of DPCM systems - particular application to voice signals', BSTJ, vol. 45, No.7, pp 1123-1151, Sept. 1966.
- [5] M.D. Paez and T.H. Glisson, 'Minimum mean squared quantization in speech', IEEE Trans. Comm. vol. COM-20, pp 225-230, April 1972.
- [6] N.S. Jayant, 'Digital coding of speech waveforms: PCM, DPCM and DM quantizers', Proc. IEEE, vol. 62, pp 611-632, May 1974.
- [7] J. Makhoul and M. Berouti, 'Adaptive noise spectral shaping and entropy coding in predictive coding of speech', IEEE Trans. Acoust. Speech and Signal Processing, vol. ASSP 27, pp 63-73, Feb. 1979.
- [8] B.S. Atal and M.R. Schroeder, 'Predictive coding of speech signals and subjective error criteria', in Conf. Rec. IEEE Int. Conf. Acoust. Speech and Signal Processing,

- [9] J.L. Flanagan et al., 'Speech coding', IEEE Trans. Commun. vol. COM-27, pp 710-737, April 1979.
- [10] P. Noll, 'A comparative study of various quantization schemes for speech encoding', BSTJ, vol. 54, pp 1597-1614, Nov. 1975.
- [11] N.S. Jayant, 'Pitch-adaptive DPCM coding of speech with two bit quantization and fixed spectrum prediction', BSTJ, vol. 56, pp 439-454, Mar. 1977.
- [12] J.D. Gibson, 'Adaptive prediction in speech differential encoding systems', Proc. IEEE, vol. 68, pp 488-525, April 1980.
- [13] P. Noll, 'Adaptive quantizing in speech coding systems', Proc. of Int. Zurich Seminar on Digital Commun. 1974,
- [14] L.R. Rabiner and R.W. Schafer, Digital Processing of Speech Signals, Prentice-Hall Inc., 1978.
- [15] N.S. Jayant, 'Adaptive quantization with one word memory', BSTJ, vol. 52, No.7, pp 1119-1144, Sept. 1973.
- [16] P. Cummiskey et al., 'Adaptive quantization in differential PCM coding of speech', BSTJ, vol. 52, No.7, pp 1105-1118, Sept. 1973.
- [17] B.S. Atal and M.R. Schoeder, 'Adaptive predictive coding of speech signals', BSTJ, vol. 49, No.8, pp 1973-1986, Oct. 1970.



P. Cummiskey, 'Adaptive differential pulse code modulation for speech processing', Ph.D. Dissertation, Newark College of Engg., Newark, N.J., 1973.

L.S. Moye, 'Self adaptive filter predictive coding system', Proc. of Int. Zurich Seminar Int. Systems for Speech, Video and Data Communications, 1972, Paper No.F3.

F. Itakura and S. Saito, 'Analysis synthesis telephony based upon the maximum likelihood method', Report of the 6th Int. Cong. Acoust., (ed.), by Y. Kohasi, Tokyo, C-5-5, 1968.

B.S. Atal and S.L. Hanauer, 'Speech analysis and synthesis by linear prediction of the speech wave', J. Acoust. Soc. Am., 50, pp 637-655, 1971.

R. Viswanathan and J. Makhoul, 'Quantization properties of transmission parameters in linear predictive systems', IEEE Trans. on Acoust. Speech and Signal Processing, vol. ASSP-23, No.3, pp 309-321, June 1975.

B.S. Atal, 'Predictive coding of speech at low bit rates', IEEE Trans. Commun., vol. COM-30, No.4, pp 600-615, April, 1982.

## CHAPTER 5

### VARIABLE LENGTH HUFFMAN CODING

#### INTRODUCTION

The input sample after being processed in the differential feedback loop, is encoded in the binary format. A normal binary code word has a fixed length of  $n$  bits/sample. The length of the code word remains the same irrespective of the probability of occupancy of the quantizer levels. Normally, it is noticed that the levels of the quantizer around zero are occupied more often than the levels near the boundaries of the quantizer. So, it looks logical to use a code such that the frequently occurring levels are encoded by a shorter length code word and the less frequent levels by longer lengths. This gives an overall average message length which is less than that given by a fixed length binary code. That is, for the same output SNR, a reduction in final data rate is achieved. Alternatively, the data rate remaining the same, an increase in SNR is possible. This of course, is achieved at the expense of a sophisticated coder with provisions of a buffer to transmit at a uniform rate in the channel and facilities to check buffer over flow/under flow conditions. Also to be included in the coder measures to combat the overflow/underflow problem of the buffer.

Historically the variable length code was not actively used in communication fields upto early 1950's, though typical example of the variable length code - the Morse Code was invented long before. The fixed length code for complete decoding needs certain conditions to be fulfilled. If  $u_1, u_2, \dots, u_L$  denote a sequence of  $L$  consecutive letters from a discrete source selected such that each letter is from an alphabet of  $a_1, a_2, \dots, a_k$ , then the number of different sequences of length  $L$  is  $k^L$ . If the code alphabet contains  $D$  symbols and of the length of each code word is  $N$ , then there are  $D^N$  different sequences of code letters available as code words. Thus, if a separate code word is to be provided for each source sequence (so that complete decoding is possible), then the condition to be satisfied is

$$\frac{N}{L} \geq \frac{\log k}{\log D} \quad (5.1)$$

So, at least  $\log k / \log D$  code letters per source letter is needed for a fixed length code. To quote an example, in teletype the source has an alphabet of  $k = 32$  symbols. Encoding single source digits ( $L = 1$ ) into binary code letters ( $D = 2$ ) needs  $N = 5$  binary digits per source symbol for errorless complete decoding [1].

Suppose the  $k$  letter alphabet  $(a_1, a_2, \dots, a_k)$  has probabilities  $p(a_1), p(a_2), \dots, p(a_k)$ . Each source letter is

to be represented by a code word consisting of a sequence of letters from a prescribed code alphabet. Let  $D$  denotes the different symbols in the code alphabet and  $n_k$  denote the number of letters in the code word corresponding to  $a_k$ . The average number of code letters per source letter is  $\bar{n}$  and

$$\bar{n} = \sum_{k_1=1}^k p(a_{k_1}) n_{k_1} \quad (5.2)$$

From law of large numbers [2], by encoding a very long sequence of source letters, the number of code letters per source letter will be close to  $\bar{n}$ , with high probability.

## 5.2 HUFFMAN CODE

A constructive procedure for finding an optimum set of code words to represent a given set of messages was discovered by D.A. Huffman [3]. By 'optimum', it is meant that no other uniquely decodable set of code words has a smaller average code word length than the given set. In this thesis, only binary codes are considered.

Let the source letters  $a_1, a_2, \dots, a_k$  have probabilities  $p(a_1), p(a_2), \dots, p(a_k)$ . It is assumed for simplicity, that the letters are ordered so that  $p(a_1) \geq p(a_2) \geq \dots \geq p(a_k)$ . The code words  $X_1, X_2, \dots, X_K$  corresponding to an optimum code and the lengths  $n_1, n_2, \dots, n_K$  often would not be unique. But by imposing certain conditions that are to be satisfied by at-least one optimum code, a procedure is arrived upon to construct this code.

Huffman gave the following as some of the properties of optimum variable length codes.

- i) No two code words would consist of an identical arrangement of code letters.
- ii) The code words are determined in such a way that no additional indication is necessary for decoding, once the starting of the sequence of code words is given
- iii) No source letter be coded in such a way that its code word is a prefix of any other code word or that many of its prefixes are used elsewhere for another source letter. Because of this property, this variable length code word is also known as prefix condition code.
- iv) If several source letters have equal probabilities (i.e.)  $p(a_1) = p(a_2) = p(a_3)$  and so on, it is possible that code words determined for these source letters may differ in length. But these code words can be interchanged without the average length getting altered.

With the above properties, condition 1 is stated as follows  
 For any given source with  $k \geq 2$  letters, an optimum binary code exists in which the two least likely code words  $X_k$  and  $X_{k-1}$  have identical lengths and differ in only the last code digit. The proof of this is given in [1]. With this condition, the problem of constructing an optimum code is reduced to that of constructing  $X_1, X_2, \dots, X_{k-2}$  and finding the first  $(n_{k-1})$  digits of  $X_k$ . Now the source has letters  $a'_1, a'_2, \dots, a'_{k-1}$  with

the probabilities

$$p_r(a'_{k_1}) = \begin{cases} p(a_{k_1}) & k_1 \leq k-2 \\ p(a_{k-1}) + p(a_k) & k_1 = k-1 \end{cases} \quad (5.3)$$

That is the source has reduced ensemble now. The problem of finding an optimum code is to find an optimum code for this reduced ensemble of source letters. Again, the least two probable messages are grouped and a further reduced ensemble is generated. Continuing in this way, a point is reached when the ensemble has only two messages and then an optimum code is clearly designed by assigning '1' to one message and '0' to another.

### 5.3 CODE CONSTRUCTION AND BUFFER MANAGEMENT

The quantizer output is the source consisting of  $2^N$  different letters. The probability of these letters are to be determined. This is done by taking a computer count to determine the frequencies of occurrence of different levels for 3 and 4 bit quantizers, and are given in the results at the end of this chapter. With the probabilities known, as explained in Section 2, an optimum code is constructed [4]. The average number of code letters per source letter is calculated. This when multiplied by the sampling rate gives the final data rate achieved with the optimum code.

The variable length code is obviously to be sent at a uniform rate on the channel. So, a buffer is required in which the samples at the input are of variable rate and transmitted on to the channel at uniform rate. This might result in situations wherein the buffer might get full or totally empty. Both these situations are to be avoided. So, an effective buffer management is necessary [5],[6].

To combat buffer overflow conditions, an adaptive strategy is resorted to. The Huffman coder has two modes of operation. (i) buffer normal and (ii) buffer full. In the normal mode, the quantizer levels or  $2^N$  letters are encoded into the respective Huffman code words. When the buffer is full, a buffer status signal causes the quantizer and the Huffman coder to switch over to operation with lesser number of levels. Typically a quantizer and Huffman coder operating with 16 levels under buffer normal conditions, would switch to 6 levels under buffer full situation. So, the rest of the levels are combined to map into the reduced number of levels [5].

To operate the Huffman coder in the adaptive mode primary considerations are the buffer length and the buffer gap. The buffer gap is that many number of bits which separates the buffer full condition from buffer normal condition. For example, let the buffer length be 'r' bits. When the buffer

contents reach this ' $r$ ' bits, the buffer is full and a buffer status signal causes the system to switch to other mode of operation with lesser number of levels. This reduces the average input to the buffer and causes the buffer to gradually empty. When the buffer contents are reduced to  $(r-r_1)$  bits, the buffer normal status is declared and original operation with higher number of levels is restored. The quantity  $r_1$  bits is what is known as buffer gap, an important parameter [7].

The number of samples in the receive buffer plus those in the transmit buffer must always be a constant. They represent the delay incurred by employing variable length coding. When the transmit buffer is full, receive buffer is empty and vice versa. Provided there are no transmission errors, the status of the transmit buffer is always known by the receiver. No buffer status signals need be sent. The single exception is when the transmit buffer is empty. In this case, the buffer must send a dummy signal to mark time until it has enough information to send.



STATISTICS OF PROBABILITY OF OCCURRENCE OF 3 BIT  
QUANTIZER LEVELS

No.	Quantizer level	Prob. of occurrence	Huffman code
1	-0.222	0.413	0
2	00.222	0.299	10
3	-0.785	0.116	110
4	0.785	0.114	1110
5	1.576	0.037	11110
6	2.994	0.012	111110
7	-1.576	0.009	1111110
8	-2.994	0.000	1111111

The No. of bits/sample with variable length Huffman code  
is 2.135

---

# STATISTICS OF PROBABILITY OF OCCURRENCE OF 4 BIT QUANTIZER LEVELS

No.	Quantizer level	Prob.of occurrence	Huffman code
1	-0.126	0.245	00
2	0.126	0.215	11
3	-0.407	0.186	011
4	0.407	0.125	0101
5	-0.726	0.074	1000
6	0.726	0.067	1010
7	1.095	0.034	10010
8	1.540	0.018	10011
9	-1.095	0.016	01000
10	2.103	0.007	101100
11	2.895	0.007	010010
12	-1.540	0.004	101110
13	4.316	0.002	101111
14	-2.103	0.000	101101
15	-2.895	0.000	0100111
16	-4.316	0.000	0100110

The No. of bits/sample with variable rate Huffman codes is 2.992

## REFERENCES

- [1] R.G. Gallager, Information Theory and Reliable Communication, New York, John Wiley and Sons, 1968.
- [2] A. Papoulis, Probability, Random Variables and Stochastic Processes, New York, McGraw-Hill, 1965.
- [3] D.A. Huffman, 'A method for the construction of minimum redundancy codes', Proc. IRE, vol. 40, pp 1098-1101, September, 1952.
- [4] M.C. Chow, 'Variable length redundancy removal coders for differentially coded video telephone signals', IEEE Trans. Commun. vol. COM-21, pp 706-714, June 1973.
- [5] K. Goyal and J.B. O'Neal Jr., 'Entropy coded differential pulse code modulation systems for television', IEEE Trans. Commun. vol. COM-23, pp 660-666, June 1975.
- [6] K. Virupaksha and J.B. O'Neal, Jr., 'Entropy coded adaptive differential pulse code modulation (DPCM) for speech', IEEE Trans. Commun. vol. COM-22, pp 777-787, June 1974.
- [7] Z.L. Budrikis et al., 'Transient-mode buffer stores for nonuniform code TV', IEEE Trans. Commun. Technol. vol. COM-19, pp 913-922, Dec. 1971.

## CONCLUSION

## 6.1 CONCLUDING REMARKS

An adaptive predictive coding scheme using maximum entropy method is studied in this thesis. The results are obtained for a first order, second order and third order predictors. Three or four bit quantization is employed. The quantizer and the predictor are adaptively exercised. It is found that the SNR improvement resulting from quantizer adaptation is very nearly 5 dB. This, together with 5-6 dB improvement in SNR due to the differential mode operation, results in an overall 10-11 dB improvement in SNR over non-adaptive PCM, for the same final data rate. The SNR improvement due to the adaptive operation of predictor is 1 dB.

For the first order predictor (ADPCM first order), it is noticed that the SNR is 19.6 dB. This figure is nearly 1.2 dB more than the figure quoted by N.S. Jayant [1] for the same data rate. The improvement can be attributed to a better prediction strategy, resulting from a more accurate determination of the autocorrelation coefficient using MEM. For a third order predictor the SNR obtained is 21.07 dB for data rate of 24 Kb/s which compares favourably with the 21.0 dB quoted by P. Noll [2]. It is noticed that the saturation value

of SNR for order of predictor three and beyond, is the same as already quoted in the literature but the rate of growth of SNR versus the order of the predictor is fast. This makes the present system, fit enough to be studied in hardware, where mostly first order predictors are used.

The concept of 'local stationarity' is revealed in the results. For block from samples 700-800, it is found, for a 4 bit quantizer with adaptive operation of quantizer and predictor, when the block length is 100 samples, that the SNRB for the block is 13.08. When the block size is reduced to 50 it is found that the SNRB for block of samples 700-750 is 22.76 dB and for block of samples 750-800 is 15.57 dB. So, the block SNR is 19.1 dB. This clearly shows that, as the input speech samples show much variations in block of 700-800, a better prediction by imposing the stationarity locally for 50 samples duration is achieved.

By employing variable length Huffman coding for 4 bit quantization, the final data rate is reduced, for the same output SNR, from 24 Kb/s to 17.95 Kb/s, resulting in a reduction of the transmission bandwidth needed by 25 percent. Similarly, for 3 bit quantization the reduction is from 18 Kb/s to 12.81 Kb/s. The bandwidth necessary to transmit the coefficient to the receiver, is an additional 1-2 Kb/s.

Subjective evaluation has not been done. But it is generally accepted that in the evaluation of a system, like the one dealt in this thesis, subjective analysis plays a useful role.

In the next section, we present some suggestions about the possible future avenues of work pertaining to this thesis.

## 6.2 SUGGESTIONS FOR FUTURE WORK

It is already stated that subjective perception of speech does not depend on the MS value of the quantization error alone. The spectral shaping of quantization noise is important. The noise in the formant regions (frequency regions where speech energy is concentrated) is masked to a large degree by signal itself. So, the frequency components of noise around the formant regions can be allowed to have a higher energy. This is done by a prefilter in the coder and a post filter in the decoder. It is shown by [3] that the spectrum of quantizing noise in the reconstructed speech signal is inversely proportional to  $[1-R(w)]^{-2}$  where  $[1-R(w)]$  is the fourier transform of the prefilter impulse response. So, any desired spectrum of noise can be achieved to attain a good subjective performance. This problem is worth exploring further. The necessary material is found in [4],[5],[6],[7].

The second possible avenue of future study that stems out of this thesis is motivated by results of rate distortion theory.

The basic idea is that if a slight delay can be tolerated at the transmitter, then the speech coder can 'look ahead' at several possible paths and hence achieve a better fit to the incoming waveform than the single path performed by DPCM. This concept of delayed encoding has actually come from the results of 'tree' coders [8]. It is shown that [9],[10] tree codes can perform close to the rate distortion bound. Recent papers on this topic [11], [12] promise exciting studies in this field.

The concept of variable frame rate transmission has seen rapid developments recently. In the case of feed forward adaptation procedures, the parameters are quantized and sent to the receiver once in every update interval, also known as frame. In the variable frame rate method, parameters are transmitted only, when their values have changed sufficiently over the interval since their previous transmission. The parameters for the untransmitted frames are regenerated at the receiver through linear interpolation between the parameters of the two adjacent frames. Thus, parameters transmission occur more frequently when speech characteristics is changing rapidly, as in phoneme transitions and less frequently when the speech characteristics are relatively constant. It is claimed that variable frame rate transmission can yield substantial lower transmission rates [13]. The connected literature is available in [14],[15],[16]. This area deserves attention.

The variable length Huffman coding, though simple and elegant has the associated disadvantage of buffer management. The buffer overflow/underflow problems are normally difficult to handle and involves the use of adaptation as mentioned in Chapter 5. A viable alternative is fixed length 'permutation code' [17]. It is shown [18] that optimum amplitude quantization and permutation encoding are equivalent in the sense that their rate versus distortion performances are identical. Especially for small values of distortion, the buffer size in the case of variable length code becomes very large resulting in a large average coding delay. Though the block length 'n' of the permutation code becomes high under this condition, it approaches the minimum entropy of the optimum quantizer [19]. The study of permutation codes in lieu of variable length Huffman codes is also to be looked into. The literature on this subject is found in [20], [21].



## REFERENCES

- [1] N.S. Jayant, 'Digital coding of speech waveforms PCM, DPCM and DM quantizers', Proc. IEEE, vol. 62, pp 611-632, May, 1974.
- [2] P. Noll, 'Adaptive quantizing in speech coding systems', Proc. of the 1974 Int. Zurich Seminar on Digital Communication, March, 1974.
- [3] B.S. Atal, 'Predictive coding of speech at low bit rates', IEEE Trans. Commun., vol. COM-30, No. 4, pp 600-614, Apr. '82.
- [4] P. Noll, 'On predictive quantizing scheme', BSTJ, vol. 57, pp 1499-1532, May/June, 1978.
- [5] E.G. Kimme and F.F. Kuo, 'Synthesis of optimum filters for a feedback quantization system', IEEE Trans. Circuit theory, vol. CT-10, pp 405-413, Sept. 1963.
- [6] J. Makhouli and M. Berouti, 'Adaptive noise spectral shaping and entropy coding in predictive coding of speech', IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-27, pp 63-73, Feb. 1979.
- [7] N.S. Jayant, 'Adaptive post filtering of ADPCM speech', BSTJ, vol. 60, pp 707-717, May/June, 1981.
- [8] J.B. Anderson and J.B. Bodie, 'Tree encoding of speech', IEEE Trans. Inform. Theory, vol. IT-21, pp 379-387, July, 1975.

- [9] F. Jelinek, 'Tree encoding of memoryless time discrete sources with a fidelity criterion', IEEE Trans. Inform. Theory, vol. IT-15, pp 584-590, September, 1969.
- [10] C.R. Davis and M.E. Hellman, 'On tree coding with a fidelity criterion', IEEE Trans. Inform. Theory, vol. IT-21, pp 373-378, July 1975.
- [11] H.G. Fehn and P. Noll, 'Multipath search decoding of stationary signals with applications to speech', IEEE Trans. Commun., vol. COM-30, No.4, pp 687-702, April, 1982.
- [12] L.C. Stewart et al., 'The design of Trellis waveform coders', IEEE Trans. Commun., vol. COM-30, pp 702-711, April, 1982.
- [13] R. Viswanathan, et al., 'Variable frame rate transmission: A review of methodology and application to narrow band LPC speech coding', IEEE Trans. Commun. vol. COM-30, No.4, April, 1982.
- [14] A.W.F. Huggins et al., 'Speech quality testing of some variable frame rate (VFR) linear predictive (LPC) vocoders', J. Acoust. Soc. Amer., vol. 62, pp 430-434, Aug. 1977.
- [15] R. Viswanathan et al., 'Variable frame rate narrowband speech transmission over fixed rate noisy channels', in Proc. EASCON'77 Washington DC, Sept. 1977, Paper 23.
- [16] E. Blackman et al., 'Narrowband LPC speech transmission over noisy channels', Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, Washington DC, April 1978,

- [17] D. Slepian, 'Permutation modulation', Proc. IEEE vol. 53, pp 228-236, March 1965.
- [18] T. Berger, 'Optimum quantizers and permutation codes', IEEE Trans. Inform. Theory, vol. IT-18, pp 759-765, Nov. 1972.
- [19] -----, 'Minimum entropy quantizers and permutation codes', IEEE Trans. Inform. Theory, vol. IT-28, No.2, pp 149-157, March 1982.
- [20] ----- et al., 'Permutation codes for sources', IEEE Trans. Inform. Theory, vol. IT-18, pp 160-169, Jan. 1972.
- [21] S.A. Townes, 'Permutation coding of speech', Ph.D. dissertation, Dept. of Elec. Engg., North Carolina State Univ., Raleigh, NC, 1981.

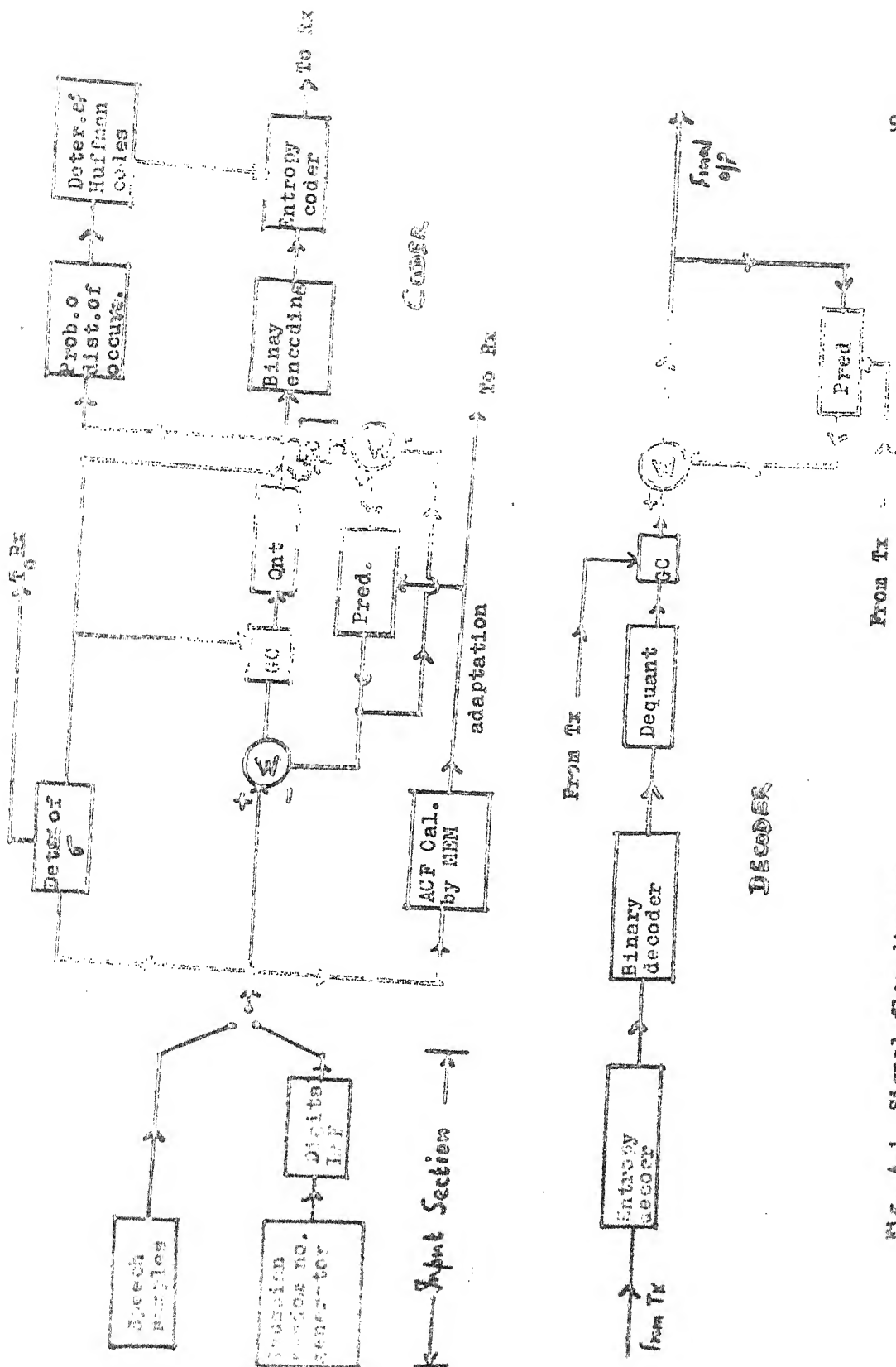


Fig. A.1 Signal flow diagram

## APPENDIX A

### A.1 SIGNAL FLOW DESCRIPTION

The signal flow diagram of the simulation experiment is shown in Fig. A.1. The figure comprises the coder and decoder sections. The input section of the coder produces speech samples or Gaussian samples. These samples are processed in the feedback loop comprising the quantizer and predictor. Both of them are made adaptive by the adaptation strategies explained in Chapter 4. The output of the quantizer is then subjected to a counting operation to determine the probability distribution statistics of the occupation of different quantizer levels. This facilitates the construction of the variable length Huffman codes. Binary encoding is done on the output of the quantizer and is further encoded in the entropy coder.

Exactly corresponding inverse operations are accomplished in the decoder side. The final reconstructed signal is available at the output.

### A2.1 GAUSSIAN NUMBER GENERATION

In a computer (like DEC-10 of IIT/K) random number sequences are generated easily. The random numbers generated have an orderly randomness in them and so they are called as 'pseudo random' sequence of numbers.

Gaussian random numbers with mean  $\mu$  and variance  $\sigma^2$  can be generated by initially generating two uniformly distributed random numbers  $r_1$  and  $r_2$ . These numbers are then converted to the Gaussian distributed random number  $s$  by the Box-Muller transformation. The Gaussian random number  $s$  is

$$s = (-2 \log_e r_1)^{\frac{1}{2}} \cos(2\pi r_2) \quad (\text{A.1})$$

From the sample  $s$ , a sample with specified mean  $\mu$  and variance  $\sigma^2$  can be generated as [1]

$$x = \sigma s + \mu \quad (\text{A.2})$$

where  $x$  is the Gaussian random number with mean  $\mu$  and variance  $\sigma^2$ .

In the simulation experiment, the IMSL routines of the DEC-10 system are used to generate  $r_1$  and  $r_2$  which are uniformly distributed and then equations (A.1) and (A.2) are applied.

## A2.2 SPEECH SAMPLES

The speech samples are obtained from 6 KHz sampling. The bandwidth is 2940 Hz. The sentence used is 'Have you seen Bill'? The digital speech samples are from the Hewlett-Packard's FFT analyzer of EE Department, Indian Institute of Science, Bangalore.

### A.3 DIGITAL FILTER

The signal flow diagram shows two kinds of inputs :

i) speech, ii) Gaussian. The Gaussian samples are to be bandlimited before feeding them to the system input. Also, the decoder output is to be filtered to get back the signal. These filterings are achieved by a digital low pass filter.

The filter realization can be done in a number of ways. It is called recursive, if the present output sample is a function of past outputs as well as past and present inputs. A filter is nonrecursive when the present output sample is a function of only the past and present inputs. Resorting to  $z$  transform notation, a digital recursive filter can be written as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^N a_i z^{-i}}{\sum_{i=0}^N b_i z^{-i}} ; b_0 = 1 \quad (A.3)$$

where  $X(z)$  is the  $z$  transform of the input,  $Y(z)$  is the  $z$ -transformer output and  $H(z)$  is the  $z$ -transform of system function. In the above equation  $a_i$ 's and  $b_i$ 's are constants to be determined.

The above equation (A.3) can be rewritten as a difference equation by cross multiplying terms as

$$Y(z) \sum_{i=0}^N b_i z^{-i} = X(z) \sum_{i=0}^N a_i z^{-i} \quad (A.4)$$

or

$$\sum_{i=0}^N b_i z^{-i} yY(z) = \sum_{i=0}^N a_i z^{-i} X(z) \quad (\text{A.5})$$

The term  $z^{-i} y(z)$  is the  $z^{-1}$  transform of  $y(n-i)$ . Taking  $z^{-1}$  of equation (A.5) leads to

$$\sum_{i=0}^N b_i Y(n-i) = \sum_{i=0}^N a_i X(n-i) \quad (\text{A.6})$$

Expanding the LHS of the above equation (A.6)

$$b_0 y_n + b_1 y(n-1) + \dots = \sum_{i=0}^N a_i X(n-i) \quad (\text{A.7})$$

or

$$Y_n = \sum_{i=0}^N a_i X(n-i) - \sum_{i=1}^N b_i Y(n-i) \quad (\text{A.8})$$

Since  $b_0 = 1$ . If  $b_0 \neq 1$ , then

$$Y_n = \sum_{i=0}^N (a_i/b_0) X(n-i) - \sum_{i=1}^N (b_i/b_0) Y(n-i) \quad (\text{A.9})$$

The structure realized by the above equation (A.9) is called the direct form 1 [2]

A filter with system function  $H(z)$  can be realized as a cascade of a number of individual lower order filters. The  $H(z)$  can be written as

$$H(z) = a_i \prod_{i=1}^K H_i(z) \quad (\text{A.10})$$



where each individual  $H_i(z)$  can be a first order section like

$$H_i(z) = \frac{1 + a_{1i}z^{-1}}{1 + b_{1i}z^{-1}} \quad (\text{A.11})$$

or a second order section like

$$H_i(z) = \frac{1 + a_{1i}z^{-1} + a_{2i}z^{-2}}{1 + b_{1i}z^{-1} + b_{2i}z^{-2}} \quad (\text{A.12})$$

$K$  is known as the order of the overall cascaded filter. The realizations of individual  $H_i(z)$  can be in direct form 1.

In general the overall  $H(z)$  of the cascaded filter can be written as

$$H(z) = \prod_{i=1}^K a_i \left[ \frac{a_{1i} + a_{2i}z^{-1} + a_{3i}z^{-2}}{b_{1i} + b_{2i}z^{-1} + b_{3i}z^{-2}} \right] \quad (\text{A.13})$$

The filter designed in this thesis is of the above type. This requires the knowledge of the following parameters [3].

- i) Determination of  $K$ , the order of the cascaded filter  
(number of individual filters in the cascade)
- ii) The denominator of  $H(z)$
- iii) The numerator of  $H(z)$

The following are the filter parameters, necessary for the design :

- i)  $w_{c1}$  - the lower cutoff frequency                      2940 Hz
- ii)  $w_{c2}$  - the upper cutoff frequency                      2990 Hz

iii)	$w_s$ - the sampling frequency	6000 Hz
iv)	$\rho_1$ - the ripple in the passband	0.05 dB
v)	$\rho_2$ - the ripple in the stopband	0.01 dB
vi)	$\tau$ - the transition ratio	0.983

The design of the filter is followed up in flow charts. The first flowchart 'FILORD' determines  $K$ . This is depicted in Fig. A.2. Starting with some prescribed values calculated from FILORD, the second flowchart 'POZE' determines the poles and zeros of the system function. POZE is shown in Fig. A.3. The next flowchart 'FILCON', calculates the coefficients  $a_{1i}$ ,  $a_{2i}$ ,  $a_{3i}$ ,  $b_{1i}$ ,  $b_{2i}$  and  $b_{3i}$  of individual  $H_i$ 's for all  $i = 1, \dots, K$ , (Fig. A.4). Also calculated are the  $a_i$ 's. With the coefficients and  $K$  known, the final flowchart 'SAMFIL' (Fig. A.5) takes the input sample, does the filtering and the filtered sample is given at the output. To begin with, the initial conditions are assumed to be zero. When a sequence is given at the input, it is the flowchart SAMFIL which is repeatedly applied as the coefficients are calculated once for all at the beginning using the other flowcharts.

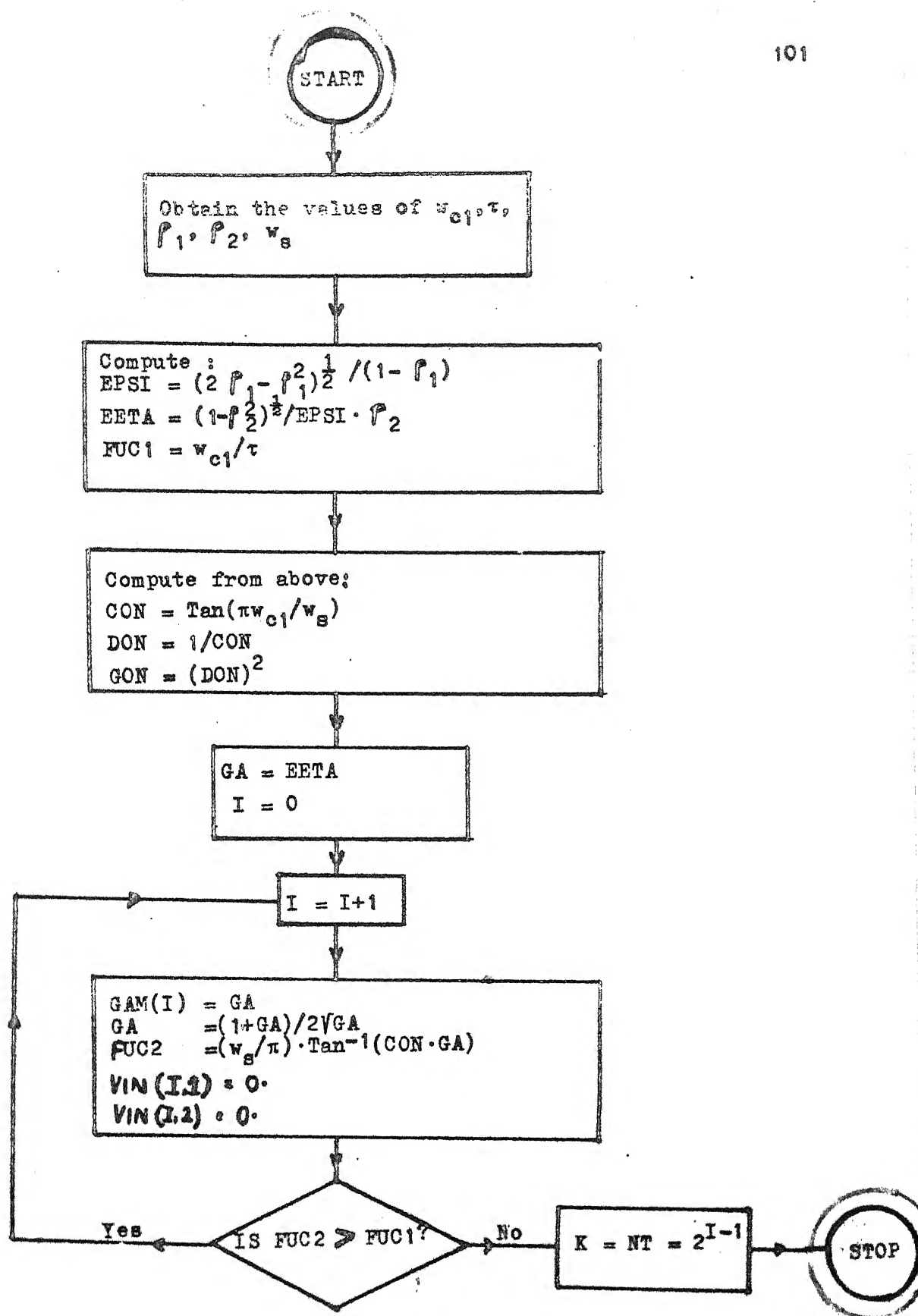


Fig. A.2 Flow chart (FILORD)

Start with prescribed values of  
EPSI, GAM and NT

Compute:  
 $Zero(1) = \left( \frac{1+GAM(1)}{2} \right)^{1/2}$   
 $Pole(1) = \left[ \frac{(1+GAM(1)) \cdot (1+EPSI/2)}{1+GAM(1)+EPSI/2} \right]^{1/2}$

Yes  
 IS NT = 1?

A

No

I = 1;  
J = 1

B

I = I + 1

NJIG =  $2^{I-2}$ 

J = J + 1

NG = NJIG + J

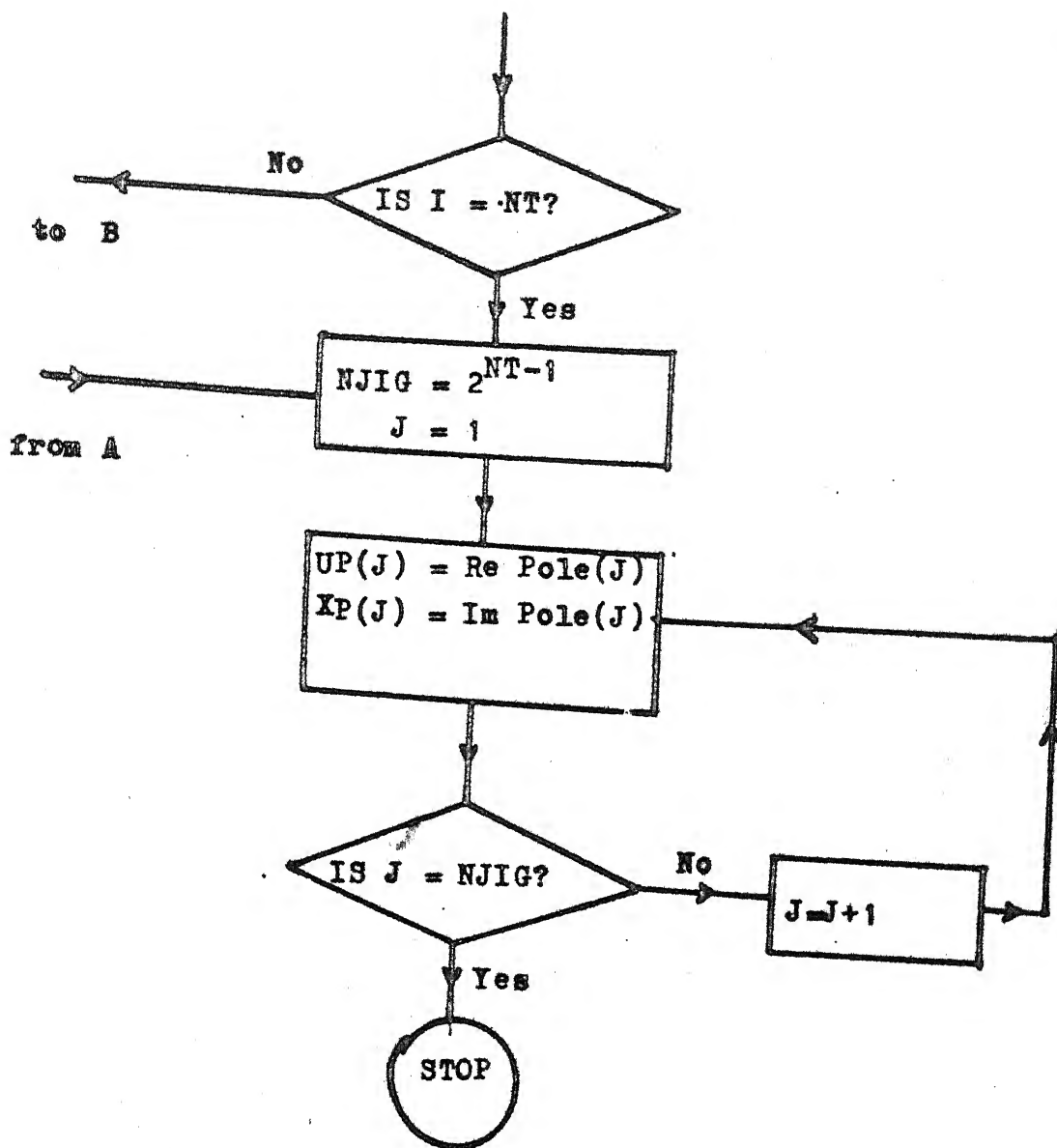
Compute:  
 Zero(NG); Pole(NG)  
 Zero(J); Pole(J)

No

IS J = NJIG?

Yes

contd...



**Fig. A.3 Flow chart (POZE)**

Start with prescribed values of DON, GON, NT  
and matrices ZERO, UP, XP

I = 0

I = I + 1

J = 1

Compute: (Numerator constants)

$$A(I, J, 1) = GON + [ZERO(I)]^2$$

$$A(I, J, 2) = -2[GON - (ZERO(I))^2]$$

$$A(I, J, 3) = A(I, J, 1)$$

$$DD = UP(I)^2 + XP(I)^2$$

$$EE = 2 \cdot XP(I) \cdot DON$$

J = J + 1

Compute: (Denominator constants)

$$A(I, J, 1) = DD + EE + GON$$

$$A(I, J, 2) = 2 \cdot (DD - GON)$$

$$A(I, J, 3) = DD - EE + GON$$

contd ...

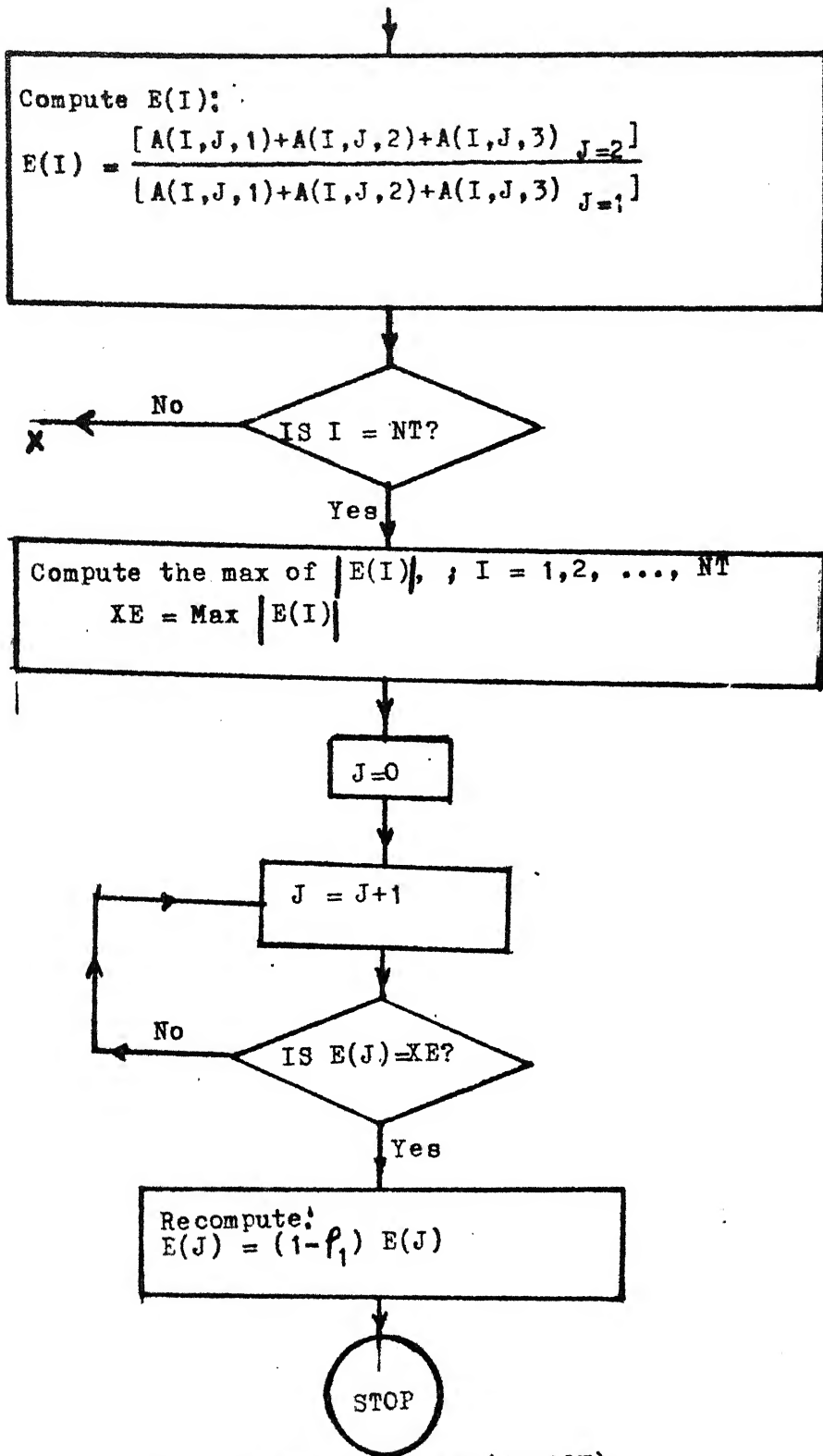


Fig. A. 4 Flow chart (FILCON)

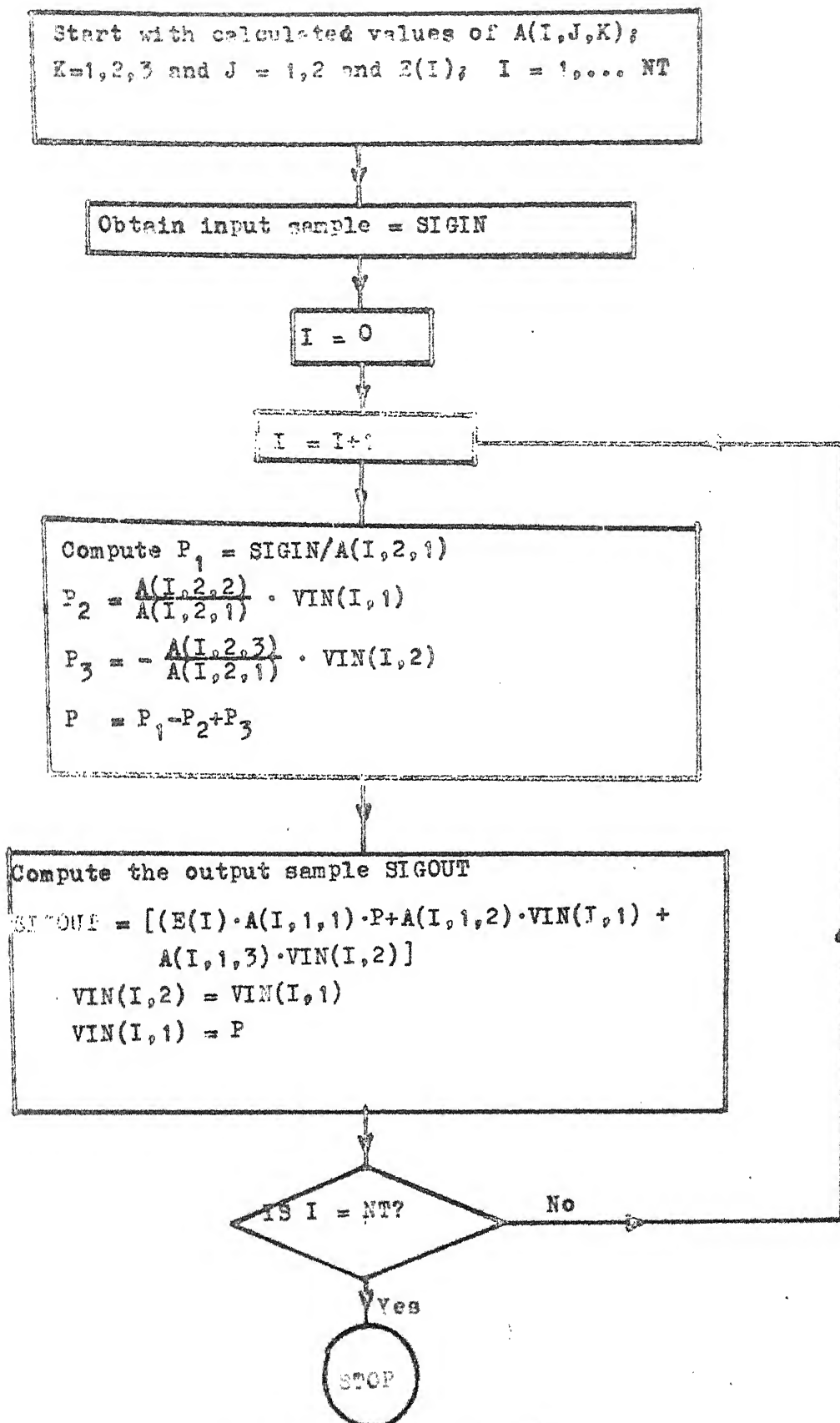


Fig. A.6 Flow chart (SAMMYL)



## APPENDIX B

The following pages from 108-138 contain the FORTRAN listing of the simulation. The listing starts with the main program. The subroutines then follow. The various routines are explained as and when they occur.

This is the main program of the simulation project  
"Adaptive predictive coding using VPM algorithm".  
The various routines along with their I/O parameters  
are explained as they occur in the program. In the  
main program the following choices/selections are  
to be made.

Whether speech or gaussian samples are to be used as input;  
Whether 3 or 4 bit quantisation is to be used;  
Whether the quantiser is to be made adaptive;  
Whether the predictor is to be made adaptive;  
Whether binary encoding/decoding is to be employed;  
If binary coding, whether variable length entropy  
coding/decoding is to be employed;

DIMENSION F(20),A(20,2,3),GAA(20),ZEP0(20),UP(20),XP(20)  
DIMENSION R1(1000),R2(1000),S(1000),SIGID(1000),VIN(20,2)  
DIMENSION VGR(1000),ACE(250,250),ACEC(250),A(250),X1(250)  
DIMENSION PRF(250),B(250),PPE(250),G(250),PER(250)  
DIMENSION XEDBK(100),ERR(100),QNT(100),QNTIP(100)  
DIMENSION DECOU(100),SAP(50),COUNT(16),IOUT(4),ITPAUS(100,4)  
DIMENSION TENTRO(7),IFITPA(100,7),DECIN(100),NDSIN(100)  
DIMENSION ROSOUT(100),CI(20,2)  
DIMENSION ERROR(20),CURFF(20),ALPHA(20,20),BETA(20),ZFETA(20)  
DIMENSION QND(20,100),SAMIN(20,100),SAMOUT(20,100)  
REAL ROSIN,ROSOUT  
INTEGER QNTSIZ

DATA EMU,SIGMA/0.0,1.0/  
DATA R1,R2,TR,FS,FLC/0.05,0.01,0.98,6000.0,2940.0/  
TYPE 49

FORMAT(4X,'OPERATOR TO NOTE:YOU ARE REQUIRED TO KEYIN VARIOUS  
DATA,AS AND WHEN NOTIFIED ON THE TERMINAL.THIS IS DONE BY  
KEYINGIN THOSE DATA FOUND WITHIN " ",AS REQUIRED.')  
TYPE 85

FORMAT(4X,'PLEASE KEYIN THE VALUE OF QNTSIZ')  
TYPE 31

FORMAT(4X,'KEYIN "3" FOR 3 BIT QUANTISER ;OR "4" FOR 4 BIT  
QUANTISER')  
ACCEPT\*,QNTSIZ

IA=1000  
I=100

IXORD=4  
IL=10

DO 13 I=1,16

COUNT(I)=0.0

CONTINUE  
TYPE 88

FORMAT(4X,'NOW PLEASE KEYIN THE VALUE OF ISELB3')  
TYPE 27

FORMAT(4X,'KEYIN "0" FOR SPEECH ;OR KEYIN "1" FOR GAUSSIAN  
SAMPLES')  
ACCEPT\*,ISELB3

CALL PRF(FLC,TR,R1,R2,FS,E,A,NT)

DO 13 I=1,16

DO 17 J=1,2

VIN(I,J)=0.0

CIN(I,J)=0.0

CONTINUE

IF(ISELB3.EQ.0)GO TO 83

OPEN(UNIT=1,FILE='GAUSS.COR')

READ(1,100)SIGID(I),I=1,IA

COUNT(I)=1,1000='GAUSS.COR')



```

IF (ISELE5.EQ.0) GO TO 93
CALL DEVAL3(M, IFTIPA, ITRAMS)
CONTINUE
CALL DECF13(ITRAMS, M, DECI13)
GO TO 75
CONTINUE
IF (ISELE5.EQ.0) GO TO 94
CALL DEVAL4(M, IFTIRA, ITRAMIS)
CONTINUE
CALL DECF14(ITRAMS, M, DECI14)
GO TO 75
CONTINUE
CALL DECODE(QNT, M, X, MAXORD, DECODE, SEGSTD, ISELE1)
GO TO 79
CONTINUE
CALL DECODE(DEC13, M, X, MAXORD, DECODE, SEGSTD, ISELE1)
CONTINUE
K1=(LI-1)*100+1
K2=LI+100
WRITE(1,9)(K1,K2)
FORMAT(8X, 'THE FOLLOWING RESULTS ARE FOR THE SEGMENT OF SAMPLES
1 FROM', I4, ' TO ', I4)
WRITE(1,21)
FORMAT(8X, '*****')
WRITE(1,7)
FORMAT(8X, 'INPUT SAMPLE', 4X, 'QNTSR INPUT', 4X, 'QNTSR OUTPUT', 4X,
1 'ERROR SIGNAL', 4X, 'DECODED OUTPUT')
WRITE(1,11)
FORMAT(8X, '*****', 4X, '*****', 4X, '*****', 4X,
1 '*****')
QVATD1=0.0
SUMM=0.0
DO 1000 I=1,M
SUMM=SUMM+WGN(I)*WGN(I)
IF (ISELE1.EQ.0) GO TO 172
WGN(I)=WGN(I)*SEGSTD
CONTINUE
QNT(I,I)=ERR(I)
SAMPIN(I,I)=WGN(I)
SAMPIN(I,I)=DECODE(I)
WRITE(1,16)(I, WGN(I), QNTIP(I), QNT(I), EPR(I), DECODE(I))
FORMAT(13, 4X, F12.6, 4X, F11.6, 4X, F12.6, 4X, F12.6, 4X, F12.6/)
WGN(I)=WGN(I)-DECODE(I)
QVAR=QNTIP(I)**2
QVATOT=QVATOT+QVAR
1000 CONTINUE
CALL MOSFLL(A, E, CIN, NT, M, NOSIN, SUMQD)
SUMQD=SUMQD/FLOAT(M)
SNRLNG=10.0*LOG10(SUMSIG/SUMQD)
TYPE1, SNRLNG, A, E, CIN, SUMQD
WRITE(1,19)(' ')
FORMAT(13, 'THE SNR FOR THE ABOVE SEGMENT IS:', F12.6/)
QVATOT=QVATOT/FLOAT(M)
GAIN=10.0*LOG10(SUMSIG/QVATOT)
WRITE(1,29)(' ')
FORMAT(13, 'THE GAIN OVER PCM IS:', F12.6/)
1 SNRLNG=SNRLNG
41 WRITE(1,32)(' ')
FORMAT(13, 'THE OVERALL SNR CALCULATION FOR THE ENTIRE INTERVAL')

```

```

45 WRITE(1,43)
   FORMAT(6X,'*****')
47 WRITE(1,47)
   FORMAT(6X,'SEGMENT NO',10X,'SNR')
51 WRITE(1,51)
   FORMAT(6X,'*****',10X,'**',/)
   SUMSNR=0.0
   DO 59 I=1,ML
     WRITE(1,53)(I,SNR(I))
     FORMAT(10X,I2,10X,F12.6,/)
     SUMSNR=SUMSNR+SNR(I)
59 CONTINUE
   QUESNR=SUMSNR/FLOAT(ML)
   WRITE(1,61)(QUESNR)
   FORMAT(12X,'THE OVERALL SNR FOR THE ENTIRE INTERVAL IS:',F12.6,/)
   CLOSE(UNIT=1,FILE='ABC.FOR')
   OPEN(UNIT=1,FILE='ABCD.FOR')
   WRITE(1,63)
   FORMAT(10X,'THE VARIOUS OCCUPANCY RATES OF THE QUANTISER')
   WRITE(1,67)
   FORMAT(10X,'*****')
   WRITE(1,69)(COUNT(I),I=1,16)
   FORMAT(30X,F12.6,/)
   CLOSE(UNIT=1,FILE='ABCD.FOR')
   OPEN(UNIT=1,FILE='ERROR.FOR')
   WRITE(1,*)(COND(IJ,I),I=1,M),IJ=1,ML)
   OPEN(UNIT=1,FILE='INPUT.FOR')
   WRITE(1,*)(SAMIN(IJ,I),I=1,M),IJ=1,ML)
   OPEN(UNIT=1,FILE='OUTPUT.FOR')
   WRITE(1,*)(SAMOUT(IJ,I),I=1,M),IJ=1,ML)
   TYPE 39
39 FORMAT(4X,'OPERATOR TO NOTE:THE PROGRAM EXECUTION IS OVER.FOR
1 DETAILED RESULTS OF THE ANALYSIS PLEASE REFER TO THE FILE
1 "ABC.FOR AND "ABCD.FOR".')
STOP
END

```

The Digital filter routine comprises of

i. Subroutine DIGFIL

ii. Function XMAX

iii. Subroutine DIGCAL

This subroutine "DIGFIL" is the filter routine which is used to bandlimit the input. The filter is a LPF with the following input-output parameters.

INPUTS: R1-is the ripple in the pass band.

R2-is the ripple in the stop band.

TR-is the transition ratio.

FS-is the sampling frequency.

FLC-is the lower cutoff frequency.

OUTPUTS: NT-is the order of the designed filter.

E-is a one dimensional array that receives the constants Eq of the equation.

A-is a three dimensional array with  $A(I,I,K) = a_{ik}$  and  $A(I,2,K) = b_{ik}$  where a's and b's are the coeffs.

The routine starts here.

```

The flow chart "FILUPU" begins from now.
SUBROUTINE DIGFIL (FIC, (F,P1,K2,FS,E,A,NT)
COMMON POLE(20),GUM
DIMENSION E(20),A(20,2,3),GA*(21),ZERO(20),JP(20),XP(20)
TYPE*,R1,R2,TR,FS,FLC
EPSI=SQRT((A1**2.)-(R1**2.))/(1.-R1)
ETA=SQRT((1.-R2**2.)/(EPSI**2.))
FUC1=FLC/FS
CU=SIGN(3.14159*FUC/FS)/COS(3.14159*FUC/FS)
TYPE*,EPSI,ETA,FUC1,CU
CU=1./CU
GUM=CU**2.009
GA=ETA
I=0
I=I+1
GA*(I)=GA
TYPE*,GA
GA=(1.+GA)/(2.*SQRT(GA))
FUC2=(FS/3.14159)*ATAN(CU*GA)
IF(FUC2.GT.FUC1) GO TO 10
NI=(2**I)/2
The flow chart "FILUPU" ends here.
The flow chart "PUZE" begins from here.
The following portion determines the poles and zeros
of the transfer function.
X=1./EPSI
ZERO(1)=SQRT((1.+GA*(1))/2.)
POLE(1)=CMPLX(0.,X)
TYPE*,ZERO(1),POLE(1)
X=ZERO(1)
GUM=CMPLX(X,0.)
X=GA*(1)
POLE(1)=GUM*CSQRT((CMPLX(1.,0.)+POLE(1))/(CMPLX(X,0.)+POLE(1)))
TYPE*,POLE(1)
IF(NT.EQ.1) GO TO 20
DO 15 I=2,NI
NIIG=(2**I)/4
X=GA*(I)
GUM=CMPLX(X,0.)
DO 15 J=1,NIIG
IG=NIIG+J
TYPE*,ZERO(J)
ZERO(IG)=SQRT(((1.+X)/2.)*(1.-ZERO(J))/(X-ZERO(J)))
ZERO(J)=SQRT(((1.+X)/2.)*(1.+ZERO(J))/(X+ZERO(J)))
TYPE*,ZERO(J),ZERO(J)
POLE(IG)=CSQRT((CMPLX(1.,0.)+GUM)*(CMPLX(1.,0.)-POLE(J))
1/(CMPLX(2.,0.)*(GUM-POLE(J))))
POLE(J)=CSQRT((CMPLX(1.,0.)+GUM)*(CMPLX(1.,0.)+POLE(J))
1/(CMPLX(2.,0.)*(GUM+POLE(J))))
CONTINUE
NIIG=(2**NT)/2
DO 25 J=1,NIIG
UP(J)=REAL(POLE(J))
XP(J)=IMAG(POLE(J))
CONTINUE
The flow chart "PUZE" ends here.
The flow chart "FILCO" begins from here. This includes the
"PUZE" flow also.
The following portion determines the filter coeffs.
DO 30 I=1,NI
A(I,1,1)=(1.-ZERO(I))**2

```

```

A(I,1,2)=-2.*(GDN-ZERO(I)**2)
A(I,1,3)=A(I,1,1)
DD=DD(I)**2+XP(I)**2
P=2.*ABS(XP(I))*DD.
A(I,2,1)=DD+EE+GDN
A(I,2,2)=2.*(DD-GDN)
A(I,2,3)=DD-EE+GDN
E(I)=(A(I,2,1)+A(I,2,2)+A(I,2,3))
1/(A(I,1,1)+A(I,1,2)+A(I,1,3))
TYPE 29,E(I)
FORMAT('E(I)',F12.6)
CONTINUE
I=0
XE=XXMAX(E,NT)
I=I+1
IF(E(I),EQ,XE) GO TO 40
GO TO 35
E(I)=E(I)*(I.-R1)
TYPE*,E(I)
RETURN
END

```

```

FUNCTION XMAX(C,I)
DIMENSION C(20)
XMAX=0.0
DO 50 I=1,L
IF(ABS(XMAX)-ABS(C(I)).GE.0.0)GO TO 50
XMAX=ABS(C(I))
TYPE 49,XMAX
FORMAT('XMAX',F12.6)
CONTINUE
RETURN
END

```

The flow chart "FILCON" ends here.

```

The flow chart "SAMFIL" starts from here.
SUBROUTINE DIGCAL(A,E,VIN,NT,MAM,SIGIN,WGN)
DIMENSION A(20,2,3),E(20),VIN(20,2),SIGIN(1000),WGN(1000)
TYPE 53,V
FORMAT('V IS',F12.6)
DO 65 J=1,MAM
V=SIGIN(J)
DO 60 I=1,NT
TYPE*,A(I,2,1)
P1=V/A(I,2,1)
P2=(A(I,2,2)/A(I,2,1))*VIN(1,1)
P3=-(A(I,2,3)/A(I,2,1))*VIN(1,2)
TYPE*,P1,P2,P3
P=(P1/A(I,2,1))-(A(I,2,2)/A(I,2,1))*VIN(1,1)
1-(A(I,2,3)/A(I,2,1))*VIN(1,2)

```

```

1  TYPE 55,P
2 55  FWRITE('D 15',F12,6)
3  V=FC1)*(A(I,1,1)*P+A(I,1,2)*VIN(I,1)
4  1+A(I,1,3)*VIN(I,2))
5  VIN(I,2)=VIN(I,1)
6  VIN(I,1)=P
7  CONTINUE
8  IG(I)=V
9  TYPE*,SIGIN(I),WGN(I)
10 55  CONTINUE
11  RETURN
12  END

```

```

1  SUBROUTINE ADAPT(WGN,SUMSIG,M,SEGSTD)
2  INPUTS: WGN-Input samples before adaption
3  SUMSIG-denotes the total energy in the segment
4  M-No. of input samples in the segment
5  SEGSTD- The standard deviation of the segment
6  OUTPUTS: WGN-Output samples after thro' the amplifier with
7  gain of 1/(SEGSTD)
8  DIMENSION WGN(3000)
9  SEGSTD=SQRT(SUMSIG)
10  DO 49 I=1,M
11  WGN(I)=WGN(I)/SEGSTD
12  CONTINUE
13  RETURN
14  END

```

```

1  SUBROUTINE ACFCEN(WGN,M,MAXORD,ACFC,FPE,G)
2  INPUTS: WGN-The input samples to cal. the autocorrelation coeffs
3  M-The no. of samples in the segment
4  MAXORD-The max. order of the Prediction filter of the
5  MEM algorithm
6  OUTPUTS: ACFC-The calculated autocorrelation coeffs
7  FPE-Akaike's final prediction error
8  G-The filter coeffs. of the prediction filter
9  DIMENSION WGN(3000),ACFC(250),FPE(250),G(250),PER(250)
10  DIMENSION PEF(250),H(250)
11  SUM=0.0
12  DO 115 I=1,M
13  SUM=SUM+WGN(I)*WGN(I)
14  CONTINUE
15  ACFC(1)=SUM/FLOAT(M)
16  PM=ACFC(1)
17  DM=ACFC(1)
18  FPE(1)=FPE(1)+FPE(1)/FPEAT(M+1)/FPEAT(M-1)*PM
19  FPE(1)=FPE(1)
20  FPE(1)=0.0

```



```

      DO 120 NN=2,(MAXORD )
      I=NN-2
      IF(N.EQ.0) GO TO 120
      DO 118 J=1,N
      PER(J)=0.0
      PER(J)=0.0
118  CONTINUE
120  SUM=0.0
      SDEM=0.0
      IJ=N-1
      DO 125 J=1,IJ
      SUM=SUM-2.0*(WGA(J+1)+PER(J))*(WGN(J)+PER(J))
      SDEM=SDEM+(WGN(J+1)+PER(J))*2+(WGN(J)+PER(J))*2
125  CONTINUE
      G(NN)=SUM/SDEM
      IF(N.EQ.0) GO TO 140
      DO 130 J=1,N
      K=N-J+2
      H(J+1)=G(J+1)+G(NN)*G(K)
130  CONTINUE
      DO 135 J=1,N
      G(J+1)=H(J+1)
135  CONTINUE
      IJ=IJ-1
140  DO 150 J=1,IJ
      PER(J)=PER(J)+G(NN)*PER(J)+G(NN)*WGN(J+1)
      PER(J)=PER(J+1)+G(NN)*PER(J+1)+G(NN)*WGN(J+1)
150  CONTINUE
      SUM=0.0
      DO 155 J=2,NN
      SUM=SUM-ACFC(NN+1-J)*G(J)
155  CONTINUE
      ACFC(NN)=SUM
      DM=(1.0-G(NN))*2)*DM
      PM=DM
      IF(NN.EQ.0) GO TO 100
      FPE(NN)=FLOAT(M+NN)/FLOAT(M-NN)*PM
      FPE(NN)=FPE(NN)/FTEMP
      FPE(NN)=ALOG10(FPE(NN))
      TYPE*,ACFC(NN)
100  CONTINUE
      DO 160 J=MAXORD,(MAXORD+1)
      SUM=0.0
      DO 170 I=2,(MAXORD-1)
      SUM=SUM-ACFC(J+1-I)*G(I)
170  CONTINUE
      ACFC(J)=SUM
160  CONTINUE
      DO 101 I=2,(MAXORD )
      ACFC(I)=ACFC(I)/ACFC(1)
101  CONTINUE
      ACFC(1)=1.0
      RETLE=
      END

```

```

SUBROUTINE ROWARRV(ACFC,MAXORD,ACF)
  INPUTS:ACFC-The autocorrelation coeffs from the MPM algorithm
  OUTPUTS:MAXORD-The max.order of the autocorrelation matrix
  OUTPUTS:ACF-The autocorrelation matrix in the Toeplitz form
  DIMENSION ACFC(250),ACF(250,250)
  The routine puts a row vector(ACFC(1)) into
  an array(ACF(L,N)).
  L=1
  K=2
  I=MAXORD-1
  DO 250 I=1,N
    M=K-1
    ACF(L,M)=ACFC(1)
    IF(M.EQ.2)GO TO 210
    GO TO 215
  210 DO 220 J=2,M
    ACF(L,J)=ACFC(J)
  220 CONTINUE
    GO TO 280
  215 M=K-2
    DO 225 II=2,(K-1)
    ACF(L,M)=ACFC(II)
    MM=MM+1
  225 CONTINUE
    II=K
    DO 230 II=2,(N-K+2)
    ACF(L,MM)=ACFC(II)
    MM=MM+1
  230 CONTINUE
  240 L=L+1
    K=K+1
  250 CONTINUE
    JJ=2
    DO 290 I=1,N
    ACF(I,(N+1))=ACFC(JJ)
    JJ=JJ+1
  290 CONTINUE
  RETURN
END

```

```

SUBROUTINE LINEQN(ACF,MAXORD,X)
  This routine finds the solutions of a system of linear
  equations by the Gauss elimination method
  INPUTS:ACF-The autocorrelation matrix in the Toeplitz form
  MAXORD-The max.order of the autocorrelation matrix
  OUTPUTS:X-The vector X contains the solution of the system of
  linear equations.(i.e) the values of the tap gains
  of the feedback linear prediction filter
  REAL MAX,MM
  INTEGER Z
  DIMENSION ACF(250,250),BCF(250,250),X(250)
  Z=MAXORD-1
  DO 310 K=1,(Z-1)

```

```

      MAX=ABS(ACF(K,K))
      Z=K
      DO 325 M=(K+1),N
      IF(ABS(ACF(M,K)).GT.MAX)GO TO 320
      GO TO 325
      MAX=ABS(ACF(M,K))
      Z=M
325  CONTINUE
      IF(Z.EQ.K)GO TO 335

      DO 330 L=K,(N+1)
      TEMP=ACF(K,L)
      ACF(K,L)=ACF(Z,L)
      ACF(Z,L)=TEMP
330  CONTINUE
335  CONTINUE
      DO 340 I=(K+1),N
      U=ACF(I,K)/ACF(K,K)
      DO 340 J=K,(N+1)
      ACF(I,J)=ACF(I,J)-U*ACF(K,J)
340  CONTINUE
      MDL=1.0
      DO 342 K=1,N
      MDL=MDL*ACF(K,K)
342  CONTINUE
      IF(ABS(MDL).EQ.0.0)GO TO 360
      WRITE(1,344)
      FORMAT('THE TRIANGULAR MATRIX IS',/)
      WRITE(1,345)((ACF(I,J),J=1,MAXORD),I=1,(MAXORD-1))
345  FORMAT(5F8.5//)
      The back substitution starts here.
      X(N)=ACF(N,(N+1))/ACF(N,N)
      WRITE(1,348)N,X(N)
348  FORMAT('THE ROOT X',I1,'IS',F12.6,/)
      DO 355 I=1,(N-1)
      K=N-I
      SUM=0.0
      DO 350 J=(K+1),N
      SUM=SUM+ACF(K,J)*X(J)
350  CONTINUE
      X(K)=(ACF(K,(N+1))-SUM)/ACF(K,K)
      WRITE(1,352)K,X(K)
352  FORMAT('THE ROOT X',I1,'IS',F12.6,/)
355  CONTINUE
      GO TO 370
      WRITE(1,365)
365  FORMAT('THE GIVEN MATRIX IS SINGULAR')
370  CONTINUE
      RETURN
      END

```

SUBROUTINE FEDBCK(M,MAXORD,WGN,X,ERR,ONT,ONTIP,COUNT,IOUT  
 1,IIRANS,IENTRO,IPITRA,MN1,ONTSI2,ISELE4,ISELE5,SEGSTD,ISELE1)  
 This routine can be described as the heart of the system  
 It linearly predicts the present value of the sample based on

the past outputs.

INPUTS: M-The no. of the samples in the segment  
 MAXORD- here it denotes the no of past output samples,  
 (MAXORD-1) over which the prediction is based  
 WGN-The input samples of the segment  
 X-The linear prediction filter tap gains  
 QNTSIZ- Size of the Quantiser: 3 or 1 bit  
 ISELE1- Selection for quantiser adaptation  
 ISELE4- Selection for binary encoding/decoding  
 ISELE5- Selection for variable length coding/decoding  
 SEGSTD-The standard deviation of the segment

The above 3 parameters are keyed in by the operator.

OUTPUTS: ERR-The quantisation error  
 QNT-The output of the quantiser  
 QNTIP-The input to the quantiser  
 COUNT-The vector containing the quantiser occupancy  
 statistics for the entire interval  
 IOUT-is the binary encoded vector of the quantiser  
 output  
 ITRANS-IOUT expressed in 2 dimension for the segment  
 under consideration  
 IENTRO-is the variable length coded vector of the  
 binary code  
 IFITPA-IENTRO expressed in 2 dimension for the segment  
 under consideration  
 MNT-Size of the variable length code vector

DIMENSION XFDBK(100), ERR(100), WGN(3000), X(20), QNT(100), QNTIP(100),  
 DIMENSION COUNT(16), IOUT(4), ITRANS(100, 4), IENTRO(7), IFITPA(100),  
 DIMENSION ALOC(100)

INTEGER QNTSIZ, QNTFUL

IF (ISELE4.EQ.0) GO TO 507

WRITE(1, 501)

501 FORMAT(10X, 'MSB', 4X, 'MSB-1', 4X, 'LSB+1', 4X, 'LSB', 8X, 'BIT-1', 4X,  
 1'BIT-2', 4X, 'BIT-3', 4X, 'BIT-4', 4X, 'BIT-5', 4X, 'BIT-6', 4X, 'BIT-7',  
 WRITE(1, 503)

503 FORMAT(10X, '\*\*\*', 4X, '\*\*\*\*\*', 4X, '\*\*\*\*\*', 4X, '\*\*\*', 8X, '\*\*\*\*\*', 4X,  
 1'\*\*\*\*\*', 4X, '\*\*\*\*\*', 4X, '\*\*\*\*\*', 4X, '\*\*\*\*\*', 4X, '\*\*\*\*\*',  
 507 CONTINUE

IF (ISELE1.EQ.0) SEGSTD=1.0

DO 500 I=1, M

IF (I.EQ.1) GO TO 505

GO TO 510

505 CONTINUE

QNTIN=WGN(1)/SEGSTD

QNTIP(1)=QNTIN

GO TO 525

510 CONTINUE

FDBK=0.0

DO 515 J=1, (MAXORD-1)

IF ((I-J).EQ.0) GO TO 520

XFD=X(J)\*ALOC(I-J)

FDBK=FDBK+XFD

515 CONTINUE

520 CONTINUE

XFDBK(I)=FDBK

QNTIN=(WGN(1)-XFDBK(I))/SEGSTD

QNTIP(I)=QNTIN

525 CONTINUE

IF (QNTSIZ.EQ.3) GO TO 550

CALL QNTSR4(QNTIN, QNTOUT)

CALL COUNT4(QNTOUT, COUNT)

```

IF(ISELE4.EQ.0)GO TO 555
CALL ENCOD4(ONTOUT,IOUT,IENTRO,4*1)
IF(ISELE5.EQ.0)GO TO 526
WRITE(1,527)((IOUT(L),L=1,QNTSIZ),(IENTRO(J),J=1,MNI)
527 FORMAT(11X,11,7X,11,8X,11,7X,11,11X,11,8X,11,8X,11,8X,11,8X,11,
16X,11,8X,11,/)
GO TO 555
528 CONTINUE
WRITE(1,530)((IOUT(L),L=1,QNTSIZ)
530 FORMAT(11X,11,7X,11,8X,11,7X,11,/)
GO TO 555
531 CALL QNTSR3(ONTIN,ONTOUT)
CALL COUNT3(ONTOUT,COUNT)
IF(ISELE4.EQ.0)GO TO 555
CALL ENCOD3(ONTOUT,IOUT,IENTRO,4*1)
IF(ISELE5.EQ.0)GO TO 536
WRITE(1,537)((IOUT(L),L=1,QNTSIZ),(IENTRO(J),J=1,MNI)
537 FORMAT(19X,11,7X,11,8X,11,11X,11,8X,11,8X,11,8X,11,8X,11,8X,
111,8X,11,/)
GO TO 555
538 CONTINUE
WRITE(1,540)((IOUT(L),L=1,QNTSIZ)
540 FORMAT(19X,11,7X,11,8X,11,/)
555 CONTINUE

OAT(I)=QNTOUT
ERR(I)=QNTIN-ONTOUT
ADCC(I)=XFDBK(I)+(OAT(I)*SEGSTD)
IF(ISELE4.EQ.0)GO TO 500
DO 553 N=1,QNTSIZ
IPTS(I,N)=IOUT(N)
553 CONTINUE
IF(ISELE5.EQ.0)GO TO 500
DO 547 K=1,MNI
IPTRA(I,K)=IENTRO(K)
547 CONTINUE
500 CONTINUE
IF(ISELE4.EQ.0)GO TO 560
WRITE(1,533)
533 FORMAT(25X,'THE VARIOUS OCCUPANCY RATES FOR THE SEGMENT')
WRITE(1,537)
537 FORMAT(25X,'*****'/)
QNTFUL=2*QNTSIZ
WRITE(1,541)((COUNT(I),I=1,ONTFUL)
541 FORMAT(40X,F12.6/)
560 CONTINUE
RETURN
END

```

The routines that follow namely QNTSR2, QNTSR3 and QNTSR4 are the optimum quantiser designs of Pass and Clisson for input of Laplacian density. All of them have the same input output parameters  
SUBROUTINE QNTSR2(ONTIN,ONTOUT)  
This routine is for 2 bit quantisation  
INPUT:ONTIN-Input to quantiser  
OUTPUT:QNTOUT-Output of the quantiser

```

      IF((CONTIN.GE.0.).AND.(CONTIN.LT.0.9816))GO TO 410
      IF((CONTIN.GE.0.9816))GO TO 411
      IF((CONTIN.LT.0.).AND.(CONTIN.GT.(-.9816)))GO TO 412
      ONTOUT=-1.510
      GO TO 413
410  ONTOUT=0.4528
      GO TO 413
411  ONTOUT=1.510
      GO TO 413
412  ONTOUT=-0.4528
413  CONTINUE
      RETURN
      END

```

```

      SUBROUTINE QNTSR3(CONTIN,ONTOUT)
      This routine is for 3bit quantisation
      IF((CONTIN.GE.0.).AND.(CONTIN.LT.0.504))GO TO 410
      IF((CONTIN.GE.0.504).AND.(CONTIN.LT.1.181))GO TO 411
      IF((CONTIN.GE.1.181).AND.(CONTIN.LT.2.285))GO TO 412
      IF((CONTIN.GE.2.285))GO TO 413
      IF((CONTIN.LT.0.).AND.(CONTIN.GE.(-.504)))GO TO 414
      IF((CONTIN.LT.(-.504)).AND.(CONTIN.GE.(-1.181)))GO TO 415
      IF((CONTIN.LT.(-1.181)).AND.(CONTIN.GE.(-2.285)))GO TO 416
      ONTOUT=-2.994
      GO TO 417
410  ONTOUT=0.222
      GO TO 417
411  ONTOUT=0.785
      GO TO 417
412  ONTOUT=1.576
      GO TO 417
413  ONTOUT=2.994
      GO TO 417
414  ONTOUT=-.222
      GO TO 417
415  ONTOUT=-.785
      GO TO 417
416  ONTOUT=-1.576
417  CONTINUE
      RETURN
      END

```

```

      SUBROUTINE QNTSR4(CONTIN,ONTOUT)
      This routine is for 4bit quantisation
      IF((CONTIN.GE.0.).AND.(CONTIN.LT.0.266))GO TO 410
      IF((CONTIN.GE.0.266).AND.(CONTIN.LT.0.566))GO TO 411
      IF((CONTIN.GE.0.566).AND.(CONTIN.LT.0.910))GO TO 412
      IF((CONTIN.GE.0.910).AND.(CONTIN.LT.1.317))GO TO 413
      IF((CONTIN.GE.1.317).AND.(CONTIN.LT.1.821))GO TO 414
      IF((CONTIN.GE.1.821).AND.(CONTIN.LT.2.499))GO TO 415
      IF((CONTIN.GE.2.499).AND.(CONTIN.LT.3.605))GO TO 416

```

```

IF(ONTIN.GE.3.605)GO TO 417
IF((ONTIN.LT.0.0).AND.(ONTIN.GE.(-.266)))GO TO 418
IF((ONTIN.LT.(-.266)).AND.(ONTIN.GE.(-.566)))GO TO 419
IF((ONTIN.LT.(-.566)).AND.(ONTIN.GE.(-.910)))GO TO 420
IF((ONTIN.LT.(-.910)).AND.(ONTIN.GE.(-1.317)))GO TO 421
IF((ONTIN.LT.(-1.317)).AND.(ONTIN.GE.(-1.821)))GO TO 422
IF((ONTIN.LT.(-1.821)).AND.(ONTIN.GE.(-2.499)))GO TO 423
IF((ONTIN.LT.(-2.499)).AND.(ONTIN.GE.(-3.605)))GO TO 424
ONTOUT=-4.316
GO TO 425
410 ONTOUT=0.126
GO TO 425
411 ONTOUT=0.407
GO TO 425
412 ONTOUT=0.726
GO TO 425
413 ONTOUT=1.095
GO TO 425
414 ONTOUT=1.540
GO TO 425
415 ONTOUT=2.103
GO TO 425
416 ONTOUT=2.895
GO TO 425
417 ONTOUT=4.316
GO TO 425
418 ONTOUT=-.126
GO TO 425
419 ONTOUT=-.407
GO TO 425
420 ONTOUT=-.726
GO TO 425
421 ONTOUT=-1.095
GO TO 425
422 ONTOUT=-1.540
GO TO 425
423 ONTOUT=-2.103
GO TO 425
424 ONTOUT=-2.895
425 CONTINUE
RETURN
END

```

SUBROUTINE DECODE(DECIN,M,X,MAXORD,DECOU,SEGSTD,ISELE1)  
 This routine DECODE accepts the dequantised sample value  
 and reconstructs the final output based on the previous  
 outputs  
 INPUTS:DECIN-Input to the decoder,from the dequantiser  
 M-The no.of samples in the segment  
 X-Vector of tap gains of the linear prediction filter  
 MAXORD-1 -Maximum ORD-ler of the above filter  
 SEGSTD-Standard deviation of the segment  
 ISELE1-Selection keyed in by the operator for adapting  
 quantiser.  
 OUTPUT:DECOU-Final analog output of the decoder  
 DIMENSION DECIN(100),X(20),DECOU(100)

```

IF(1$ELE1.EQ.0)SEGSTD=1.0
DO 530 I=1,M
IF(1.EQ.1)GO TO 535
GO TO 540
535 CONTINUE
DECOU(I)=DECIN(I)*SEGSTD
GO TO 530
540 CONTINUE
DFOBK=0.0
DO 550 J=1,(MAXORD-1)
IF((I-J).EQ.0)GO TO 555
DXFD=X(J)*DECOU(I-J)
DFOBK=DFOBK+DXFD
550 CONTINUE
555 CONTINUE
DECOU(I)=(DECIN(I)*SEGSTD+DFOBK)
530 CONTINUE
IF(1$ELE1.EQ.0)GO TO 560
DO 560 I=1,M
DECOU(I)=DECOU(I)*SEGSTD
560 CONTINUE
RETURN
END

```

#### SUBROUTINE COUNT4(ONTOUT,COUNT)

This routine COUNT4 gives the quantiser occupancy statistics for the 4bit quantiser

INPUT:ONTOUT-Output of the quantiser

OUTPUT:COUNT-Vector containing the quantiser occupancy statistics for the entire interval

```

DIMENSION COUNT(16)
IF(ONTOUT.EQ.0.126)GO TO 610
IF(ONTOUT.EQ.0.407)GO TO 615
IF(ONTOUT.EQ.0.726)GO TO 620
IF(ONTOUT.EQ.1.095)GO TO 625
IF(ONTOUT.EQ.1.540)GO TO 630
IF(ONTOUT.EQ.2.103)GO TO 635
IF(ONTOUT.EQ.2.895)GO TO 640
IF(ONTOUT.EQ.4.316)GO TO 645
IF(ONTOUT.EQ.(-.126))GO TO 650
IF(ONTOUT.EQ.(-.407))GO TO 655
IF(ONTOUT.EQ.(-.726))GO TO 660
IF(ONTOUT.EQ.(-1.095))GO TO 665
IF(ONTOUT.EQ.(-1.540))GO TO 670
IF(ONTOUT.EQ.(-2.103))GO TO 675
IF(ONTOUT.EQ.(-2.895))GO TO 680
IF(ONTOUT.EQ.(-4.316))GO TO 685
610 COUNT(1)=COUNT(1)+1.0
GO TO 690
615 COUNT(2)=COUNT(2)+1.0
GO TO 690
620 COUNT(3)=COUNT(3)+1.0
GO TO 690
625 COUNT(4)=COUNT(4)+1.0
GO TO 690
630 COUNT(5)=COUNT(5)+1.0

```



```

635      GO TO 690
        COUNT(6)=COUNT(6)+1.0
        GO TO 690
640      COUNT(7)=COUNT(7)+1.0
        GO TO 690
645      COUNT(8)=COUNT(8)+1.0
        GO TO 690
650      COUNT(9)=COUNT(9)+1.0
        GO TO 690
655      COUNT(10)=COUNT(10)+1.0
        GO TO 690
660      COUNT(11)=COUNT(11)+1.0
        GO TO 690
665      COUNT(12)=COUNT(12)+1.0
        GO TO 690
670      COUNT(13)=COUNT(13)+1.0
        GO TO 690
675      COUNT(14)=COUNT(14)+1.0
        GO TO 690
680      COUNT(15)=COUNT(15)+1.0
        GO TO 690
685      COUNT(16)=COUNT(16)+1.0
690      CONTINUE
        RETURN
        END

```

SUBROUTINE ENCOD4(ONTOUT,IOUT,IENTRO,MNI)

This routine ENCOD4 does the binary and variable length Huffman encoding

INPUT:ONTOUT-Output of the quantiser

OUTPUTS:IOUT-is the binary encoded vector of the quantiser output

IENTRO-is the variable length coded vector of the binary code

MNI-Size of the variable length code vector

```

DIMENSION IOUT(4),IENTRO(7)
IF(ONTOUT.EQ.0.126)GO TO 710
IF(ONTOUT.EQ.0.407)GO TO 715
IF(ONTOUT.EQ.0.726)GO TO 720
IF(ONTOUT.EQ.1.095)GO TO 725
IF(ONTOUT.EQ.1.540)GO TO 730
IF(ONTOUT.EQ.2.103)GO TO 735
IF(ONTOUT.EQ.2.895)GO TO 740
IF(ONTOUT.EQ.4.316)GO TO 745
IF(ONTOUT.EQ.(-.126))GO TO 750
IF(ONTOUT.EQ.(-.407))GO TO 755
IF(ONTOUT.EQ.(-.726))GO TO 760
IF(ONTOUT.EQ.(-1.095))GO TO 765
IF(ONTOUT.EQ.(-1.540))GO TO 770
IF(ONTOUT.EQ.(-2.103))GO TO 775
IF(ONTOUT.EQ.(-2.895))GO TO 780
IF(ONTOUT.EQ.(-4.316))GO TO 785
710  IOUT(1)=1
      IOUT(2)=0
      IOUT(3)=0
      IOUT(4)=0
      CALL VALN40(IOUT,IENTRO,MNI)

```

```

715      GO TO 790
        IOUT(1)=1
        IOUT(2)=0
        IOUT(3)=0
        IOUT(4)=1
        CALL VALN41(IOUT,IENTRO,MNI)
720      GO TO 790
        IOUT(1)=1
        IOUT(2)=0
        IOUT(3)=1
        IOUT(4)=0
        CALL VALN42(IOUT,IENTRO,MNI)
725      GO TO 790
        IOUT(1)=1
        IOUT(2)=0
        IOUT(3)=1
        IOUT(4)=1
        CALL VALN43(IOUT,IENTRO,MNI)
730      GO TO 790
        IOUT(1)=1
        IOUT(2)=1
        IOUT(3)=0
        IOUT(4)=0
        CALL VALN44(IOUT,IENTRO,MNI)
735      GO TO 790
        IOUT(1)=1
        IOUT(2)=1
        IOUT(3)=0
        IOUT(4)=1
        CALL VALN45(IOUT,IENTRO,MNI)
740      GO TO 790
        IOUT(1)=1
        IOUT(2)=1
        IOUT(3)=1
        IOUT(4)=0
        CALL VALN46(IOUT,IENTRO,MNI)
745      GO TO 790
        IOUT(1)=1
        IOUT(2)=1
        IOUT(3)=1
        IOUT(4)=1
        CALL VALN47(IOUT,IENTRO,MNI)
750      GO TO 790
        IOUT(1)=0
        IOUT(2)=1
        IOUT(3)=1
        IOUT(4)=1
        CALL VALN48(IOUT,IENTRO,MNI)
755      GO TO 790
        IOUT(1)=0
        IOUT(2)=1
        IOUT(3)=1
        IOUT(4)=0
        CALL VALN49(IOUT,IENTRO,MNI)
760      GO TO 790
        IOUT(1)=0
        IOUT(2)=1
        IOUT(3)=0
        IOUT(4)=1
        CALL VALN4A(IOUT,IENTRO,MNI)
        GO TO 790

```

```

765      IOUT(1)=0
          IOUT(2)=1
          IOUT(3)=0
          IOUT(4)=0
          CALL VALN4B(IOUT,IENTRO,MNI)
          GO TO 790
770      IOUT(1)=0
          IOUT(2)=0
          IOUT(3)=1
          IOUT(4)=1
          CALL VALN4C(IOUT,IENTRO,MNI)
          GO TO 790
775      IOUT(1)=0
          IOUT(2)=0
          IOUT(3)=1
          IOUT(4)=0
          CALL VALN4D(IOUT,IENTRO,MNI)
          GO TO 790
780      IOUT(1)=0
          IOUT(2)=0
          IOUT(3)=0
          IOUT(4)=1
          CALL VALN4E(IOUT,IENTRO,MNI)
          GO TO 790
785      IOUT(1)=0
          IOUT(2)=0
          IOUT(3)=0
          IOUT(4)=0
          CALL VALN4F(IOUT,IENTRO,MNI)
790      CONTINUE
          RETURN
          END

```

SUBROUTINE DECFI4(ITRANS,M,DECIN)  
 This routine DECFI4 is the decoder which maps the binary  
 code into the respective dequantised samples  
 INPUTS: ITRANS - Binary encoded vector expressed in 2  
           dimension for the segment under consi-  
           deration

M - No. of samples in the segment

OUTPUT: DECIN - Binary decoded sample that is input to  
           the decoder

DIMENSION ITRANS(100,4), DECIN(100)

DO 800 I=1,M

IF(ITRANS(I,1).EQ.0)GO TO 815

IF(ITRANS(I,2).EQ.0)GO TO 820

IF(ITRANS(I,3).EQ.0)GO TO 825

IF(ITRANS(I,4).EQ.0)GO TO 830

DECIN(I)=4.316

GO TO 890

815 IF(ITRANS(I,2).EQ.0)GO TO 835

IF(ITRANS(I,3).EQ.0)GO TO 840

IF(ITRANS(I,4).EQ.0)GO TO 845

DECIN(I)=-0.126

GO TO 890

820 IF(ITRANS(I,3).EQ.0)GO TO 850

IF(ITRANS(I,4).EQ.0)GO TO 855

```

      825  DECIN(I)=1.095
           GO TO 890
      830  IF(1TRANS(I,4).EQ.0)GO TO 860
           DECIN(I)=2.103
           GO TO 890
      835  DECIN(I)=2.895
           GO TO 890
      840  IF(1TRANS(I,3).EQ.0)GO TO 865
           IF(1TRANS(I,4).EQ.0)GO TO 870
           DECIN(I)=-1.540
           GO TO 890
      845  IF(1TRANS(I,4).EQ.0)GO TO 875
           DECIN(I)=-0.726
           GO TO 890
      850  DECIN(I)=-0.407
           GO TO 890
      855  IF(1TRANS(I,4).EQ.0)GO TO 880
           DECIN(I)=0.407
           GO TO 890
      860  DECIN(I)=0.726
           GO TO 890
      865  DECIN(I)=1.540
           GO TO 890
      870  IF(1TRANS(I,4).EQ.0)GO TO 885
           DECIN(I)=-2.895
           GO TO 890
      875  DECIN(I)=-2.103
           GO TO 890
      880  DECIN(I)=-1.095
           GO TO 890
      885  DECIN(I)=0.126
           GO TO 890
      890  DECIN(I)=-4.316
      895  CONTINUE
      900  CONTINUE
           RETURN
           END

```

The following 16 routines are the variable length coder routines named as VALN40 to VALN4F, the last letter being in hexadecimal of 0 to 15. They have the same input output parameters. The "4" in the name of the routines indicate that they belong to 4bit quantiser

SUBROUTINE VALN40(IOUT,IENTRO,MNI)  
 INPUT: IENTRO-binary encoded vector of the quantiser output  
 OUTPUT: IOUT-variable length coded vector of the binary code

MNI-Size of the variable length code vector  
 DIMENSION IOUT(4),IENTRO(7)  
 IENTRO(1)=1  
 IENTRO(2)=1  
 IENTRO(3)=2  
 RETURN  
 END

```

SUBROUTINE VALN41(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=0
IENTRO(2)=1
IENTRO(3)=0
IENTRO(4)=1
MNI=4
RETURN
END

```

```

SUBROUTINE VALN42(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=1
IENTRO(4)=0
MNI=4
RETURN
END

```

```

SUBROUTINE VALN43(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=0
IENTRO(4)=1
IENTRO(5)=0
MNI=5
RETURN
END

```

```

SUBROUTINE VALN44(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=0
IENTRO(4)=1
IENTRO(5)=1
MNI=5
RETURN
END

```

```

SUBROUTINE VALN45(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=1

```

```

IENTRO(4)=1
IENTRO(5)=0
IENTRO(6)=0
MNI=5
RETURN
END

```

```

SUBROUTINE VALN46(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=0
IENTRO(2)=1
IENTRO(3)=0
IENTRO(4)=0
IENTRO(5)=1
IENTRO(6)=0
MNI=6
RETURN
END

```

```

SUBROUTINE VALN47(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=1
IENTRO(4)=1
IENTRO(5)=1
IENTRO(6)=1
MNI=6
RETURN
END

```

```

SUBROUTINE VALN48(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=0
IENTRO(2)=0
MNI=2
RETURN
END

```

```

SUBROUTINE VALN49(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=0
IENTRO(2)=1
IENTRO(3)=1
MNI=3
RETURN

```

END

```

SUBROUTINE VALN4A(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=0
IENTRO(4)=0
MNI=4
RETURN
END

```

```

SUBROUTINE VALN4B(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=0
IENTRO(2)=1
IENTRO(3)=0
IENTRO(4)=0
IENTRO(5)=0
MNI=5
RETURN
END

```

```

SUBROUTINE VALN4C(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=1
IENTRO(4)=1
IENTRO(5)=1
IENTRO(6)=0
MNI=6
RETURN
END

```

```

SUBROUTINE VALN4D(IOUT,IENTRO,MNI)
DIMENSION IOUT(4),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=0
IENTRO(3)=1
IENTRO(4)=1
IENTRO(5)=0
IENTRO(6)=1
MNI=6
RETURN
END

```

```

      ITRANS(1,2)=1
      ITRANS(1,3)=1
      ITRANS(1,4)=1
      GO TO 990
930  IF((IFITRA(I,4).EQ.0)GO TO 940
      ITRANS(1,1)=1
      ITRANS(1,2)=0
      ITRANS(1,3)=0
      ITRANS(1,4)=1
      GO TO 990
940  IF((IFITRA(I,5).EQ.0)GO TO 950
      IF((IFITRA(I,6).EQ.0)GO TO 955
      IF((IFITRA(I,7).EQ.0)GO TO 950
      ITRANS(1,1)=0
      ITRANS(1,2)=0
      ITRANS(1,3)=0
      ITRANS(1,4)=1
      GO TO 990
950  ITRANS(1,1)=0
      ITRANS(1,2)=1
      ITRANS(1,3)=0
      ITRANS(1,4)=0
      GO TO 990
955  ITRANS(1,1)=1
      ITRANS(1,2)=1
      ITRANS(1,3)=1
      ITRANS(1,4)=0
      GO TO 990
960  ITRANS(1,1)=0
      ITRANS(1,2)=0
      ITRANS(1,3)=0
      ITRANS(1,4)=0
      GO TO 990
915  IF((IFITRA(I,3).EQ.0)GO TO 965
      IF((IFITRA(I,4).EQ.0)GO TO 970
      IF((IFITRA(I,5).EQ.0)GO TO 985
      IF((IFITRA(I,6).EQ.0)GO TO 998
      ITRANS(1,1)=1
      ITRANS(1,2)=1
      ITRANS(1,3)=1
      ITRANS(1,4)=1
      GO TO 990
965  IF((IFITRA(I,4).EQ.0)GO TO 975
      IF((IFITRA(I,5).EQ.0)GO TO 980
      ITRANS(1,1)=1
      ITRANS(1,2)=1
      ITRANS(1,3)=0
      ITRANS(1,4)=0
      GO TO 990
970  ITRANS(1,1)=1
      ITRANS(1,2)=0
      ITRANS(1,3)=1
      ITRANS(1,4)=0
      GO TO 990
975  ITRANS(1,1)=0
      ITRANS(1,2)=1
      ITRANS(1,3)=0
      ITRANS(1,4)=1
      GO TO 990
980  ITRANS(1,1)=1
      ITRANS(1,2)=0

```



```

      ITRANS(1,3)=1
      ITRANS(1,4)=1
      GO TO 990
985  IF(IFITRA(I,6).EQ.0)GO TO 995
      ITRANS(1,1)=0
      ITRANS(1,2)=0
      ITRANS(1,3)=1
      ITRANS(1,4)=0
      GO TO 990
988  ITRANS(1,1)=0
      ITRANS(1,2)=0
      ITRANS(1,3)=1
      ITRANS(1,4)=1
      GO TO 990
995  ITRANS(1,1)=1
      ITRANS(1,2)=1
      ITRANS(1,3)=0
      ITRANS(1,4)=1
990  CONTINUE
990  CONTINUE
      RETURN
      END

```

```

SUBROUTINE COUNT3(ONTOUT,COUNT)
This routine COUNT3 gets the quantiser occupancy
statistics for the 3bit quantiser
INPUT:ONTOUT-Output of the quantiser
OUTPUT:COUNT-vector containing the quantiser occupancy
statistics
      DIMENSION COUNT(8)
      IF(ONTOUT.EQ.0.222)GO TO 1010
      IF(ONTOUT.EQ.0.785)GO TO 1015
      IF(ONTOUT.EQ.1.576)GO TO 1020
      IF(ONTOUT.EQ.2.994)GO TO 1025
      IF(ONTOUT.EQ.(-.222))GO TO 1030
      IF(ONTOUT.EQ.(-.785))GO TO 1035
      IF(ONTOUT.EQ.(-1.576))GO TO 1040
      IF(ONTOUT.EQ.(-2.994))GO TO 1045
1010  COUNT(1)=COUNT(1)+1.0
      GO TO 1090
1015  COUNT(2)=COUNT(2)+1.0
      GO TO 1090
1020  COUNT(3)=COUNT(3)+1.0
      GO TO 1090
1025  COUNT(4)=COUNT(4)+1.0
      GO TO 1090
1030  COUNT(5)=COUNT(5)+1.0
      GO TO 1090
1035  COUNT(6)=COUNT(6)+1.0
      GO TO 1090
1040  COUNT(7)=COUNT(7)+1.0
      GO TO 1090
1045  COUNT(8)=COUNT(8)+1.0
1090  CONTINUE
      RETURN
      END

```

```

SUBROUTINE ENCOD3(IOUT, IOUT, IENTRO, MNI)
DIMENSION IOUT(3), IENTRO(7)
IF(IOUT.EQ.0.222)GO TO 1110
IF(IOUT.EQ.0.785)GO TO 1115
IF(IOUT.EQ.1.576)GO TO 1120
IF(IOUT.EQ.2.994)GO TO 1125
IF(IOUT.EQ.(-.222))GO TO 1130
IF(IOUT.EQ.(-.785))GO TO 1135
IF(IOUT.EQ.(-1.576))GO TO 1140
IF(IOUT.EQ.(-2.994))GO TO 1145
1110 IOUT(1)=1
    IOUT(2)=0
    IOUT(3)=0
    CALL VALN30(IOUT, IENTRO, MNI)
    GO TO 1190
1115 IOUT(1)=1
    IOUT(2)=0
    IOUT(3)=1
    CALL VALN31(IOUT, IENTRO, MNI)
    GO TO 1190
1120 IOUT(1)=1
    IOUT(2)=1
    IOUT(3)=0
    CALL VALN32(IOUT, IENTRO, MNI)
    GO TO 1190
1125 IOUT(1)=1
    IOUT(2)=1
    IOUT(3)=1
    CALL VALN33(IOUT, IENTRO, MNI)
    GO TO 1190
1130 IOUT(1)=0
    IOUT(2)=1
    IOUT(3)=1
    CALL VALN34(IOUT, IENTRO, MNI)
    GO TO 1190
1135 IOUT(1)=0
    IOUT(2)=1
    IOUT(3)=0
    CALL VALN35(IOUT, IENTRO, MNI)
    GO TO 1190
1140 IOUT(1)=0
    IOUT(2)=0
    IOUT(3)=1
    CALL VALN36(IOUT, IENTRO, MNI)
    GO TO 1190
1145 IOUT(1)=0
    IOUT(2)=0
    IOUT(3)=0
    CALL VALN37(IOUT, IENTRO, MNI)
1190 CONTINUE
    RETURN
END

```

2 The following 3 routines are the variable length coder  
 3 routines named as VALN30 to VALN37. The "3" in their

date indicate that they refer to 3bit quantiser. They have the same input output parameters

IOPOF:IOUF-binary encoded vector of the quantiser output

OUTPUTS:IENTRO-variable length encoded vector of the binary code

MNI-Size of the variable length coded vector

SUBROUTINE VALN30(IOUF,IENTRO,MNI)

DIMENSION IOUF(3),IENTRO(7)

IENTRO(1)=1

IENTRO(2)=0

MNI=2

RETURN

END

SUBROUTINE VALN31(IOUF,IENTRO,MNI)

DIMENSION IOUF(3),IENTRO(7)

IENTRO(1)=1

IENTRO(2)=1

IENTRO(3)=1

IENTRO(4)=0

MNI=4

RETURN

END

SUBROUTINE VALN32(IOUF,IENTRO,MNI)

DIMENSION IOUF(3),IENTRO(7)

IENTRO(1)=1

IENTRO(2)=1

IENTRO(3)=1

IENTRO(4)=1

IENTRO(5)=0

MNI=5

RETURN

END

SUBROUTINE VALN33(IOUF,IENTRO,MNI)

DIMENSION IOUF(3),IENTRO(7)

IENTRO(1)=1

IENTRO(2)=1

IENTRO(3)=1

IENTRO(4)=1

IENTRO(5)=1

IENTRO(6)=0

MNI=6

RETURN

END

SUBROUTINE VALN34(IOUF,IENTRO,MNI)

```

DIMENSION IOUT(3),IENTRO(7)
IENTRO(1)=0
MNT=1
RETURN
END

```

```

SUBROUTINE VALN35(IOUT,IENTRO,MNT)
DIMENSION IOUT(3),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=1
IENTRO(3)=0
MNT=3
RETURN
END

```

```

SUBROUTINE VALN36(IOUT,IENTRO,MNT)
DIMENSION IOUT(3),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=1
IENTRO(3)=1
IENTRO(4)=1
IENTRO(5)=1
IENTRO(6)=1
IENTRO(7)=0
MNT=7
RETURN
END

```

```

SUBROUTINE VALN37(IOUT,IENTRO,MNT)
DIMENSION IOUT(3),IENTRO(7)
IENTRO(1)=1
IENTRO(2)=1
IENTRO(3)=1
IENTRO(4)=1
IENTRO(5)=1
IENTRO(6)=1
IENTRO(7)=1
MNT=7
RETURN
END

```

```

SUBROUTINE DECF13(TRANS,M,DECIN)
This routine DECF13 is the decoder pertaining to 3bit
quantiser, that maps the binary code into the respective
quantised samples
INPUT: TRANS-Binary encoded vector expressed in 2
dimension for the segment under consider-

```

```

      ation
      M-The no. of samples in the segment
      OUTPUT: DECIN-Dequantised output sample
      DIMENSION ITRANS(100,3),DECIN(100)
      DO 1200 I=1,M
      IF(ITRANS(I,1).EQ.0)GO TO 1210
      IF(ITRANS(I,2).EQ.0)GO TO 1215
      IF(ITRANS(I,3).EQ.0)GO TO 1220
      DECIN(I)=2.994
      GO TO 1290
1210 IF(ITRANS(I,2).EQ.0)GO TO 1225
      IF(ITRANS(I,3).EQ.0)GO TO 1230
      DECIN(I)=-0.222
      GO TO 1290
1215 IF(ITRANS(I,3).EQ.0)GO TO 1235
      DECIN(I)=0.785
      GO TO 1290
1220 DECIN(I)=1.576
      GO TO 1290
1225 IF(ITRANS(I,3).EQ.0)GO TO 1240
      DECIN(I)=-1.576
      GO TO 1290
1230 DECIN(I)=-0.785
      GO TO 1290
1235 DECIN(I)=0.222
      GO TO 1290
1240 DECIN(I)=-2.994
1290 CONTINUE
1200 CONTINUE
      RETURN
      END

```

```

      SUBROUTINE DEVAL3(M,IFITRA,ITRANS)
      This routine DEVAL3 is the first of the decoders for
      3bit quantiser, that maps the variable length code into
      binary code of uniform length
      INPUTS: M-The no. of samples in the segment
      IFITRA-The variable length code expressed in 2
      dimension for the segment under consideration
      OUTPUT: ITRANS-Uniform length output of IFITRA
      DIMENSION IFITRA(100,7),ITRANS(100,3)
      DO 1300 I=1,M
      IF(IFITRA(I,1).EQ.0)GO TO 1310
      IF(IFITRA(I,2).EQ.0)GO TO 1315
      IF(IFITRA(I,3).EQ.0)GO TO 1320
      IF(IFITRA(I,4).EQ.0)GO TO 1325
      IF(IFITRA(I,5).EQ.0)GO TO 1330
      IF(IFITRA(I,6).EQ.0)GO TO 1335
      IF(IFITRA(I,7).EQ.0)GO TO 1340
      ITRANS(I,1)=0
      ITRANS(I,2)=0
      ITRANS(I,3)=0
      GO TO 1390
      ITRANS(I,1)=0
      ITRANS(I,2)=1
      ITRANS(I,3)=1

```

1310



```

755  FORMAT('P IS',F12.6)
      V=E(I)*(A(I,1,1)*P+A(I,1,2)*CIN(I,1)
      1+A(I,1,3)*CIN(I,2))
      CIN(I,2)=CIN(I,1)
      CIN(I,1)=P
60  CONTINUE
      NOSOUT(J)=V
      TYPE*,NOSIN(J),NOSOUT(J)
      SUMOD=SUMOD+NOSOUT(J)**2
      CONTINUE
      RETURN
      END

```

**31358**

```

SUBROUTINE STABLE(MAXORD,ACFC)
This routine employs the Durbin's algorithm to determine
whether the filter is stable. The PARCOR coeffs. are calculated
in the process.
INPUTS: MAXORD-The maximum order of the recursion
        ACFC-The vector of autocorrelation values
DIMENSION ERROR(20),COEFF(20),ALPHA(20,20),BEETA(20),ACFC(20)
DIMENSION ZEETA(20)
ERROR(1)=ACFC(1)
DO 10 I=2,MAXORD
  RESULT=0.0
  IF(I.EQ.2)GO TO 20
  DO 20 J=2,(I-1)
    BEETA(J)=ALPHA(J,(I-1))*ACFC(I-J+1)
    RESULT=RESULT+BEETA(J)
  CONTINUE
20  COEFF(I)=- (ACFC(I)+RESULT)/ERROR(I-1)
    IF(ABS(COEFF(I).GE.1.0))GO TO 40
    DELTA=COEFF(I)
    ALPHA(I,I)=DELTA
    IF(I.EQ.2)GO TO 30
    DO 30 J=2,(I-1)
      ZEETA(J)=ALPHA(J,(I-1))+(COEFF(I)*ALPHA((I-J+1),(I-1)))
      ALPHA(J,I)=2*ZEETA(J)
    CONTINUE
30  ERROR(I)=(1.0-COEFF(I)**2)*ERROR(I-1)
    CONTINUE
    TYPE 35
35  FORMAT('THE ANALYSTS IS CONTINUED ON THE FEEDBACK LINEAR
1 PREDICTION FILTER IS STABLE WITH ALL THE ROOTS WITHIN
1 THE UNIT CIRCLE.')
    GO TO 50
40  TYPE 45
45  FORMAT('THE FEEDBACK LINEAR PREDICTION FILTER IS UNSTABLE.
1 CHECK UP ONCE AGAIN.')
    CONTINUE
50  STOP
      END

```